A Methodology for Analyzing the Performance of Authentication Protocols

ALAN HARBITTER
PEC Solutions, Inc.
and
DANIEL A. MENASCÉ
George Mason University

Performance, in terms of user response time and the consumption of processing and communications resources, is an important factor to be considered when designing authentication protocols. The mix of public key and secret key encryption algorithms typically included in these protocols makes it difficult to model performance using conventional analytical methods. In this article, we develop a validated modeling methodology to be used for analyzing authentication protocol features, and we use two examples to illustrate the methodology. In the first example, we analyze the environmental parameters that favor one proposed public-key-enabled Kerberos variant over another in the context of a large, multiple-realm network. In the second example, we propose a Kerberos variant for a mobile computing environment and analyze the performance benefits realized by introducing a proxy to offload processing and communications workload.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]: *design studies, modeling techniques*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*authentication*

General Terms: Design, Measurement, Performance, Security

Additional Key Words and Phrases: Authentication, Kerberos, mobile computing, performance modeling, proxy servers, public key cryptography

1. INTRODUCTION

To meet the challenge of protecting information and computing systems against unauthorized access, researchers and developers have focused attention on

Portions of this work have been published in preliminary form as HARBITTER, A. AND MENASCÉ, D. A. 2001a. Performance of public key-enabled Kerberos authentication in large networks. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy* (Oakland, Calif., May). IEEE Computer Society, Los Alamitos, Calif., pp. 170–183, and as HARBITTER, A. AND MENASCÉ, D. A. 2001b. The performance of public key-enabled Kerberos authentication in mobile computing applications. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8)* (Philadelphia, Pa., Nov.). ACM, New York, pp. 78–85.
Authors' addresses: A. Harbitter, Chief Technology Officer, PEC Solutions, Inc., 12750 Fair Lakes Circle, Fairfax, VA 22033; email: alan.harbitter@pec.com; D. Menascé, Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, VA 22030; email: menasce@cs.gmu.edu.

Table I. Encryption Processing Times Vary Widely According to Algorithm and Key Length

Encryption Algorithm	Key Length	Operation Time (msec)
DES	56	0.0006
Triple DES	112 (effective)	0.0016
AES (Rijndael)	128	0.0005
RSA encryption	1024	0.320
RSA decryption	1024	10.230
RSA encryption	2048	0.890
RSA decryption	2048	64.130

All results from the Crypto++ Benchmarks run on a Celeron 850-MHz processor under Windows 2000 SP 1 [Dai 1999].

authentication protocols. Authentication protocols often use multiple encryption algorithms with widely varying performance characteristics. Table I shows that the time required to decrypt a 64-byte block of data can vary from 0.0005 to 64.13 milliseconds—a factor of 10^5—depending upon the choices of algorithm and key length.

A common approach in authentication protocols is to use asymmetric (public key) cryptography to establish a symmetric (secret) session key. This approach takes advantage of the key management features of public key cryptography, but recognizes that public key cryptography algorithms consume more resources than symmetric key cryptography. The session key is used for the remainder of the user session and for most encryptions to provide efficient confidentiality or integrity services.

The authentication protocol designer faces many decisions about the use of encryption—including the application of public or secret key algorithms–that affect level of assurance and performance. Previous efforts to analyze protocol performance have emphasized benchmark and measurement methods. While these methods provide a picture of quantitative performance under specific system conditions, they are limited in their ability to support, in more general terms, the evaluation of the protocol design's performance characteristics. We present a methodology that is based on closed queuing network models—a tool for analyzing the performance of authentication protocol design.

Section 2 summarizes related work that addresses the performance of security protocols. Section 3 provides some background information about the proposed variants of the Kerberos authentication protocol (examples for our methodology will employ these variants). We present our methodology in Section 4. Sections 5 and 6 present two examples that demonstrate the application of the methodology to protocol design issues: Kerberos protocol design options are first examined in a large multiple-domain network (Section 5) and then in a mobile computing setting (Section 6). Section 7 summarizes our results and suggests directions for future work.

2. RELATED WORK

Developers of security protocols and encryption algorithms have long recognized the importance of performance. In many cases, researchers applied either

complexity analysis to evaluate algorithm performance or measurement techniques to analyze system performance [Bassham 1999; Blaze 1996]. Protocol performance has become an increasingly important topic as they are more commonly used in production and high-workload computing and networking environments.

Early work examined the impact of security protocols on network throughput, seeking to determine whether encryption calculations would put a damper on rapidly increasing data rates. Zorkadis [1994] identifies the communications performance impacts of five basic security services: authentication, access control, confidentiality, integrity, and nonrepudiation. Zorkadis begins his exploration of the impacts by constructing a simple queuing model for secure communications.

Because Kerberos was the standard network authentication protocol in the Open Software Foundation's Distributed Computing Environment (DCE) [Opengroup 1997], it has been analyzed in that context. DCE security services have been benchmarked and analyzed [Martinka et al. 1993]. The performance characteristics of Kerberos have been loosely measured in some of its pilot applications [Stallings 1994], with good results. El-Hadidi et al. [1999] used a single queue model based on M/D/1 and M/G/1 calculations to compare Kerberos performance to Diffie–Hellman and their proposed hybrid protocol. Orozco-Barbosa et al. [1998] used simulation analysis to evaluate the impacts of adding security services to a cellular wireless network. Their models showed significant performance degradation in response times when security services were added.

The predominant role of public key cryptography in electronic commerce has motivated several performance studies. Apostolopoulos et al. [1999] look at ways to reduce the impact of the private key encryption step in the Transport Layer Security (TLS) protocol—the Internet standard for performing authentication and establishing secure communications. Menascé and Almeida [2000] use analytical modeling to assess the trade-off between performance and security in e-commerce applications using protocols such as TLS and payment services such as SET. Lambert [1998] performs a high-level benchmark-supported analysis of performance improvements anticipated from the use of elliptic curve cryptography in e-commerce transactions.

Finally, in previous work, we conducted preliminary analyses of public-key-enabled and proxy-assisted Kerberos [Harbitter and Menascé 2001a, 2001b], forming the basis for the methodology and examples presented here. In contrast to the other work published in this area, our methodology offers a way to explicitly model the performance of new protocol designs that mix asymmetric and symmetric encryption. The flexibility of the closed queuing network formulation permits general observations to be made about the protocol's applicability in a variety of operational environments and can help guide design decisions.

3. BACKGROUND: PUBLIC-KEY-ENABLED AND PROXY-ASSISTED KERBEROS

Kerberos has become a mature network authentication protocol based on secret key cryptography. One typical characteristic of a mature protocol is the

existence of a wide range of proposals to extend it into uses that were not directly envisioned by its authors. This characteristic makes Kerberos a good candidate for analysis to determine how each proposed variant will perform and to motivate the proposal of new variants based on improving performance.

The following sections describe Kerberos operation and currently proposed variants of Kerberos that introduce two new elements: public key cryptography and a proxy server. Several proposals have been developed that add public key cryptography to various stages of Kerberos to make the protocol work with large user communities and Public Key Infrastructures (PKIs). The computational requirements of public key cryptography are significantly higher than those of secret key cryptography. As a result, the substitution of public key encryption algorithms for secret key algorithms impacts performance. The addition of a proxy into the Kerberos authentication can be used to off-load encryption processing from either the client or the servers. This may be valuable for a mobile computing setting in which processing, power, and communications resources are constrained. However, the proxy also introduces delays into the transaction by requiring authentication messages to be relayed between the client and the server.

3.1 The Basic Operation of Kerberos

Kerberos is a network authentication scheme based on the early work of Needham and Schroeder [1978]. Kerberos divides the world into realms, each with a single primary Key Distribution Center (KDC), back-up KDCs, application servers, and user workstations. A single realm corresponds to a community of interest with a single security policy. Many good, detailed descriptions of Kerberos protocol operation exist [Neuman and Ts'o 1994] and will not be repeated here. Briefly, the client (Alice) engages in a multiple-step authentication to obtain access to the application server (Bob). Alice must first obtain a Ticket-Granting Ticket (TGT) to a centralized Ticket-Granting Service (TGS) offered by the KDC. She uses the TGT to obtain a service ticket to Bob. She presents the service ticket to Bob and authenticates herself by demonstrating knowledge of a secret session key securely passed to her by the KDC.

A large enterprise may consist of many realms, and Alice may wish to gain access to an application server in a remote realm. To support "cross-realm" authentication, Alice's KDC and the remote KDC must have a trust relationship. This trust relationship is implemented by sharing symmetric keys between the KDCs. If such a trust relationship exists, Alice may gain access to the remote server by first requesting a ticket to the remote realm's KDC from her local KDC. She will receive a ticket (a data structure encrypted with the KDC pair's shared symmetric key) to the remote TGS. When she presents the ticket to the remote KDC, she can receive a service ticket for an application server in the remote realm.

3.2 Proposals to Add Public Key Encryption to Kerberos

Kerberos message formats are defined such that the session keys are always included in some encrypted portion of the message. As a result, Kerberos servers

do not need to store session keys or maintain a security association with each client. Kerberos is stateless; state is represented through the Kerberos tickets. Statelessness is valuable from the standpoint of robustness and scalability.

A potential limitation of Kerberos in terms of scalability is its reliance on symmetric key encryption [Ashely and Broom 1997]. Shared secrets must be established and maintained between every user and the KDC, between every application server and the KDC, and between remote KDCs. The use of public key cryptography shifts key management from the KDC to a Certification Authority (CA). Public key cryptography reduces the number of shared secrets (i.e., symmetric keys) between KDCs, servers, and users. However, the scaling merits of public-key- versus secret-key-based systems have not been definitively proven. In addition, the introduction of public key technology into Kerberos presents new key management challenges such as the reliable publishing and maintenance of public keys.

There are several current proposals for adding public key cryptography to Kerberos and hence changing the key management model. Internet drafts exist for three alternatives: (1) Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) [Tung et al. 2001], (2) Public Key Cryptography for Cross-Realm Authentication in Kerberos (PKCROSS) [Tung et al. 1998], and (3) Public Key Utilizing Tickets for Application Servers (PKTAPP) [Medvinsky et al. 1997]. PKINIT is the core specification. Both PKCROSS and PKTAPP use variations of PKINIT message types and data structures to apply public key cryptography to different Kerberos authentication steps.

PKINIT. The PKINIT Internet draft specifies that considerable message content must be added to the initial Kerberos exchanges in order to replace the user secret key authentication with public key authentication. The client may send a chain of public key certificates to establish trust between the user and the KDC and to relay the user's public key. The client must send an authenticating data structure signed with client's private key. This information is included in the Kerberos pre-authentication fields defined in the specification to support extensions to the protocol. The KDC verifies the client's identity by verifying the digital signature and replies to the client with a chain of certificates for the KDC's public key, the KDC's digital signature, and a secret key encrypted with the client's public key. The client can confirm the KDC's identity by verifying its digital signature.

Mandatory variations allow the substitution of a certificate serial number for the certificate chain (assuming the KDC already has a trusted copy of the user's certificate) and the use of Diffie–Hellman to establish a session key. PKINIT drafts have included several interesting options such as storing the user's private key on the KDC and allowing the user to generate the session key. (Allowing the user to generate the session key could change scaling characteristics.)

PKCROSS. While PKINIT addresses the issue of managing secret keys for a large number of clients, it does not address the issue of key management among a large number of realms. A logical extension of PKINIT is the use of public key encryption for multiple-realm, KDC-to-KDC authentications. This

is the subject of the PKCROSS Internet draft specification. PKCROSS picks up the multiple-realm authentication at the point at which the client has already obtained a TGT. The client may or may not have authenticated to its local KDC using PKINIT. If the client requests access to a server in a remote realm, its local KDC initiates a PKCROSS transaction with the appropriate remote KDC. With a few minor variations, the KDC-to-KDC authentication is performed using the PKINIT protocol.

One variation is that the remote KDC is responsible for generating a "special symmetric key it uses for PKCROSS requests" [Tung et al. 1998]. The local KDC can skip the explicit exchange with the remote KDC if it currently has an active, valid TGT sealed with a special symmetric key. Once the client possesses a remote TGT, it may request additional service tickets in the remote realm without involving the local KDC.

PKTAPP. In Kerberos, the KDC issues all tickets in its realm. Since most authentication transactions have to transit the KDC, it can become a performance bottleneck. Although secondary KDCs can be included in the system, they are typically used as backups in the event of a primary KDC failure. The PKTAPP Internet draft seeks to eliminate this potential bottleneck and reduce communications traffic by implementing the authentication exchange directly between the client and the application server. This variation was originally introduced as the Public-key-based Kerberos for Distributed Authentication (PKDA) [Sirbu and Chuang 1997]. PKTAPP proposes to implement PKDA using the PKINIT-specified message definitions and exchanges.

PKTAPP is a more efficient protocol than traditional Kerberos from a message exchange perspective: The client may deal directly with the application server. The AS–REQ message, the first message submitted by the client, contains the client's certificate chain and the identity of the service ticket requested. The server response, an AS–REP message, contains the server's certificate chain and a secret key encrypted with the client's public key. After authentication, the client requests an application service ticket using a Kerberos Version 5 request. The entire authentication process is reduced to a total of two message pairs.

We provide specifications of the PKCROSS and PKTAPP protocols in Appendix A. The PKINIT specification is embedded in the PKCROSS specification since it is used by PKCROSS authenticate between remote realms. In all of the public key extensions, there is no explicit requirement for advance knowledge of identity between the client and the KDC or between the two KDCs. There is no need to establish shared secrets or store a user record in a Kerberos database ahead of time. The basis for trust between these entities is the certificate chain.

These protocols substitute public key infrastructure as the management mechanism in lieu of sharing secret keys. Because of the additional processing requirements, a performance price is paid each time a public key calculation is substituted for a secret key calculation. Additionally, because public key messages are larger than the corresponding Kerberos Version 5 messages and more likely to fragment, the PKINIT draft recommends the use of TCP as the

underlying transport protocol. UDP, which has a significantly lower overhead, is the protocol commonly used for secret-key-based Kerberos implementations.

3.3 Proposals to Add a Proxy Server to Kerberos

The application of proxy servers in protocol processing has been widely studied [Zenel 1999; Fox and Gribble 1996]. The proxy can isolate the client and the server for security purposes or, in a mobile computing setting, the proxy can offload processing from the mobile platform or network. The following paragraphs review two protocols: IAKERB, an IETF draft to add a proxy to standard Kerberos [Swift et al. 2001], and Charon [Fox and Gribble 1996], a proxy-assisted version of Kerberos for mobile computing applications. Finally, to shed additional light on proxy-assisted authentication in a wireless setting, we describe features of another relevant protocol: the Wireless Application Protocol (WAP) Wireless Transport Layer Security (WTLS) [WAP 2000b].

IAKERB. There may be situations in which a proxy server is already in use in a computing network environment and the authentication protocol must accommodate it. The IAKERB IETF draft adapts Kerberos to operate when a client cannot communicate directly to a KDC and must do so through a proxy. IAKERB operates in two modes: (1) proxy and (2) minimal messages. In "proxy" mode, IAKERB specifies the protocol framing and addressing requirements, allowing the proxy to serve as a pass-through node between the client and KDC. In "minimal messages" mode, the Kerberos message formats are more highly modified to support a PKTAPP-style authentication.

Charon. The Charon protocol adapts standard Kerberos authentication to a mobile PDA (personal digital assistant) platform. Charon uses Kerberos to establish a trust relationship between a user and a proxy. The mechanism for establishing this relationship is similar to the method of establishing trust between a user and an application server. As a result, several more message exchanges are required in Charon than in the standard Kerberos protocol, where no proxy is involved.

Charon uses the same encryption algorithms (i.e., DES) on the PDA as standard Kerberos. There is no network performance advantage to using Charon rather than an unmodified Kerberos. The benefit of Charon is that it has a smaller memory footprint and it establishes a trust relationship between the PDA and the proxy. The trust relationship allows the PDA user to take advantage of the processing power of the proxy for compute-intensive operations.

The authors of Charon offer an example of potential areas for proxy assistance: retrieving e-mail via a Kerberized POP service and distilling MIME images in the messages to suit the client's display. A trusted proxy can perform these services without the risk of revealing private data to untrusted parties. The authors suggest that, following the logic arguments in Burrows et al. [1990], Charon can be proven to have the same authentication properties as Kerberos because their protocol does not change Kerberos semantics.

WAP WTLS. WAP provides a lightweight set of protocols that allow resource-constrained computing platforms, such as cell phones, to operate on a

data network, such as the Internet. Because the WAP protocols are not directly interoperable with the traditional Internet protocols, a gateway (i.e., proxy server) is required to perform protocol translation.

The WAP version of the TLS protocol is WTLS. WTLS provides options to perform both server-side and client-side authentication and certificate exchange. The final result is the establishment of a session key that can be used to securely exchange application data. WTLS resembles TLS, but is incompatible with it. As a result, if the target application server only supports TLS, the WAP proxy must perform a protocol translation from WTLS to TLS.

The level of assurance offered by WTLS has been criticized in the literature [Khare 1999], and there are several objections to its use. For example, it allows the use of weak encryption algorithms and features that make chosen-plaintext attacks and brute force attacks easier to mount. Further, in order to translate from WTLS to TLS, the WAP gateway must decrypt and re-encrypt messages transiting from the user to the target server. As a result, a potentially untrusted gateway has access to clear-text messages. There are several proprietary and proposed standard solutions aimed at closing what has been called the "WAP Gap" [Jormalainen and Laine 1999] and implementing end-to-end (i.e., mobile-client-to-target-server) security with WAP [Cylink 2000; WAP 2000a].

4. METHODOLOGY

Our methodology follows standard practices for conducting a performance modeling analysis: construct the model, validate the model, vary modeling parameters, and analyze the results. However, it is important to accommodate the unique characteristics of security protocols in implementing the details of each step. There are three high-level steps: develop a closed queuing network model that reflects encryption algorithm performance, validate the model, and conduct "what-if" analyses.

4.1 Develop a Closed Queuing Network Model that Reflects Encryption Algorithm Performance

To illustrate how the closed queuing network is constructed, consider Figure 1, which models a Kerberos cross-domain authentication. Customers circulate among the servers in the closed network and sequentially wait for service, consume processing resources, and then proceed to the next service station.

Suppose that there are two customers, Alice and Bob, circulating in the network. In the figure, both Alice and Bob are at the KDC queuing station. Alice is performing a public-key-based initial authentication. Bob is waiting to request a service ticket. The KDC is conducting a public key signature verification of Alice's PKINIT authenticator. When Bob enters the service center, the KDC will conduct a secret key decryption and encryption on his behalf to issue a service ticket. On Alice's next trip through the service center, the KDC will conduct secret key operations on her behalf to issue her a service ticket.

The queuing network model captures the delays experienced as Alice and Bob compete with other customers requesting services from the same resources—reflecting the performance characteristics of many customers executing

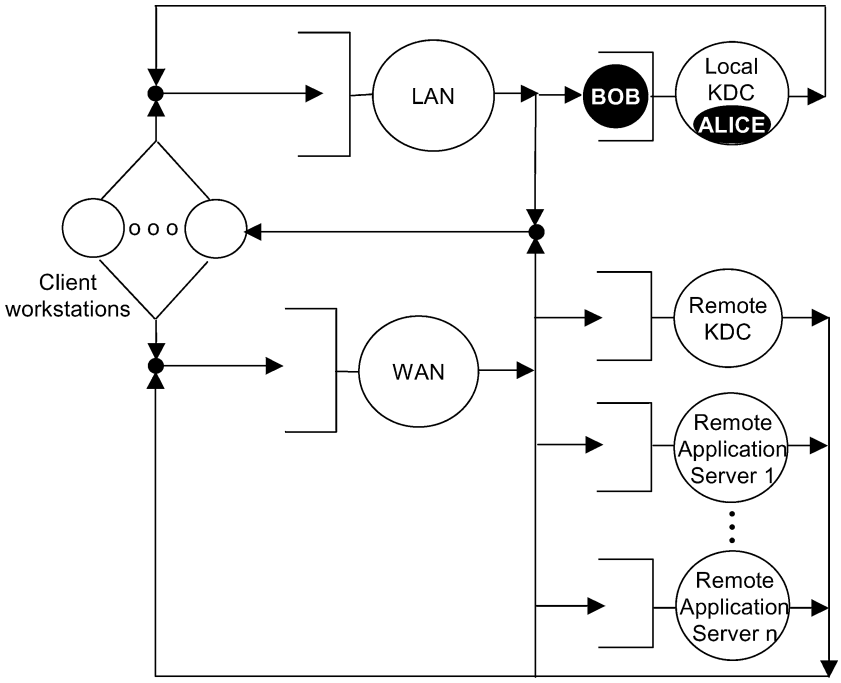

Fig. 1. Modeling topology for a multiple-realm authentication.

simultaneously as independent threads in a multitasking server. We use a class-switching formulation so the service times can be varied for each step in the protocol. When Alice leaves the KDC service center, she will switch classes, eventually joining the class Bob is in. The multiple-class, closed queuing network with class switching is well understood [Bruell and Balbo 1980]. Class switching is infrequently used in performance modeling and this is the first application to authentication protocols.

In this type of queuing network, the steady state solution retains a "product form" [Gross and Harris 1998]. In a product form queuing network, the probability that the system is in a given state $N = (n_1, n_2, \ldots, n_k)$ representing the number of customers at all k servers is proportional to the product of the marginal probabilities $P_i(n_i)$ that server i has n_i customers. The product form queuing network can be solved efficiently even for large numbers of servers and customers [Schweitzer 1991]. We provide the equations used to solve for key performance metrics in Appendix B. This formulation allows us to model a protocol with multiple phases in which each phase requires a significantly different amount of computing or communications resources than the others.

This queuing model analysis of the protocol can accurately reflect the complex performance characteristics of the authentication protocol. For example, as the number of customers increases, servers with the largest portion of the workload will saturate first. This may cause overall response times to increase nonlinearly. Queues at the saturated servers will grow faster than others in the system, resulting in underutilization of the nonsaturated server. These

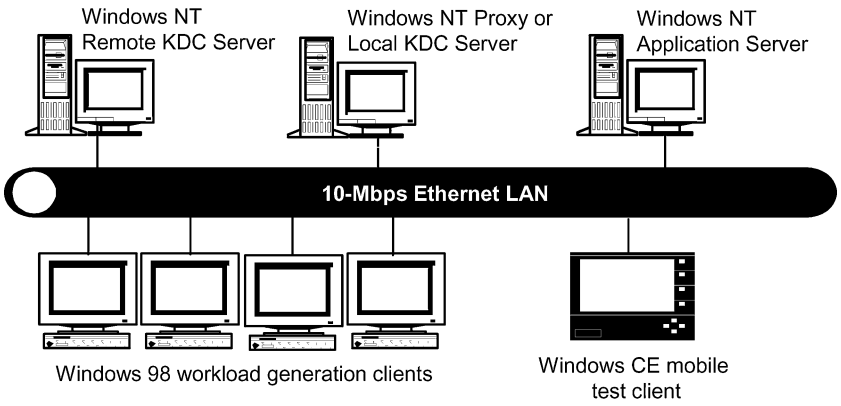

Fig. 2. Multiple-realm test bed.

effects, caused by the specific branching and service characteristics of the protocol cannot be accurately captured by a simple analysis of service times.

4.2 Validate the Model

In order to trust the model's predictions, we must demonstrate its accuracy against an implementation. It is a challenge to validate the model of a new protocol because reference implementations may not exist. To support validation of the model, we develop a "skeleton" software implementation of the protocol. The skeleton design consumes computing resources in a manner similar to actual implementation (i.e., resulting from communications, encryption, and message processing), but the skeleton software avoids many of the complexities presented by implementing the real protocols. The skeleton software includes calls to RSA and DES encryption libraries, use of TCP and UDP communications protocols, message parsing functions, and multithreading, but excludes error processing and some optional protocol features. This reduces the complexity of the software and the coding time for the skeleton, but supports validation of the model. There have been software engineering studies that generally support the rationale for our approach. Cristian [1995] finds that in operational computer software systems, often more than two-thirds of the source code is devoted to detecting and handling exceptions, yet exceptions are expected to occur infrequently.

We execute the skeleton implementation in a test bed to measure protocol no-load service times. Figure 2 illustrates our multiple-realm Kerberos test bed.

We use instrumented code, an IP-layer packet monitor, and software monitors to capture the service times and message sizes. We run automated scripts on each client workstation to generate load on the servers and measure transaction throughput and response times for several levels of workload submission. Then, we enter similar workload profiles into the queuing model to validate the model against the test bed emulations and assess predictive response time accuracy.

4.3 Conduct "What-If" Analyses

The input parameters of the validated model match the configuration constraints of the test bed and give only limited insight into operation in a production environment. We adjust these parameters to reflect conditions we would expect in the real world and explore the protocol's sensitivity to variety in the operational environment. The objective of the what-if analyses is to surface general performance characteristics of the protocol design.

5. EXAMPLE 1: THE ANALYSIS OF PUBLIC-KEY-ENABLED KERBEROS IN LARGE NETWORKS

The context for the first example is authentication in a large multiple-realm network. To illustrate a potential application for this example, consider an intelligent software agent representing a law enforcement official and collecting information for an investigation. The agent "visits" on-line town halls and virtual sheriff's offices in a nationwide search for evidence and investigative intelligence. At each stop, the agent must show well-accepted electronic credentials. The source and level of these credentials will be used to grant the agent access to records tightly controlled for reasons of confidentiality and privacy. Further, the agent may be requested to pay for information with electronic currency. The infrastructure to support this type of electronic investigation will require scalable, robust authentication protocols.

The intelligent agent will probably transit multiple security realms with varying numbers of application servers during the course of the cyber investigation. Both PKTAPP and PKCROSS are candidates for the authentication protocol. A quick analysis might conclude that PKTAPP would have better performance characteristics because the agent would authenticate directly to the application server with only two message pairs. However, the agent may be interested in authenticating to several servers within a single realm—as would be the case for a visit to the cyber town hall, courthouse, police station, and sheriff's office in the same township.

If the agent uses PKCROSS, an expensive public key authentication would be required only once—between the local KDC and the remote KDC. After the cross-authentication and the provision of a TGT to the remote TGS, only secret key encryption calculations would be required. At some application-server-to-realm ratio, it would be more efficient to use the PKCROSS protocol. This is the specific performance question we will explore: Under what circumstance is it more efficient to authenticate to a central KDC than to individual application servers?

5.1 Queuing Model

The KDCs, application server, communications networks, and client workstations are finite resources that process workload while Kerberos authentication transactions are executing. We constructed a closed queuing network model to represent each resource used by the protocols. The queuing network topology in Figure 1 is suitable to support representation of both PKCROSS and PKTAPP. The topology envisioned in Figure 1 anticipates that the local KDC may

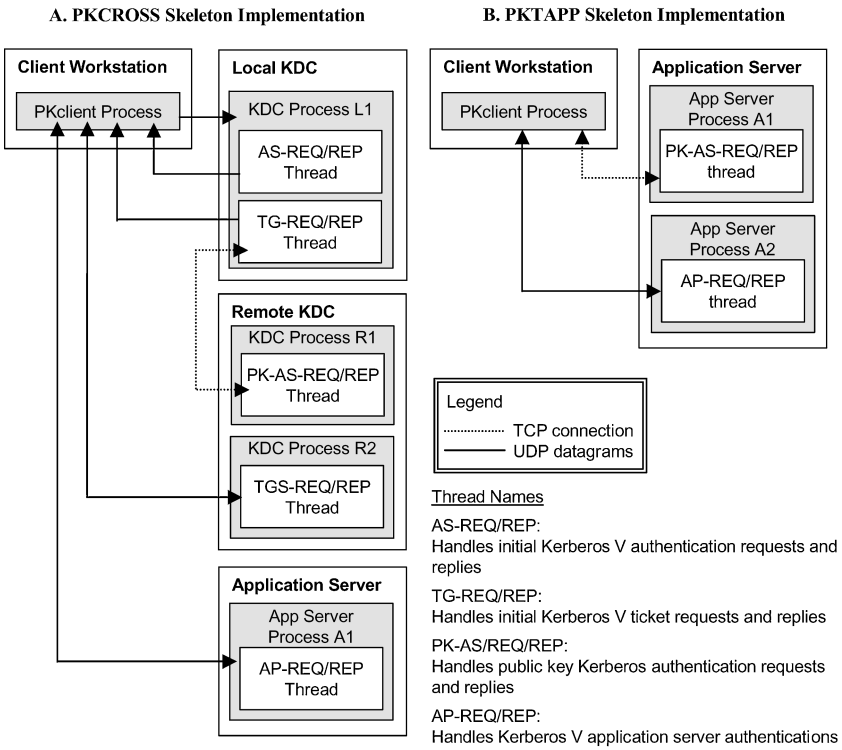

Fig. 3. The public-key-enabled Kerberos skeleton software architecture.

be connected to the client by a local area network (LAN), and the remote KDC and application server may be connected by a wide area network (WAN). The validated model will use the LAN to connect all KDCs and server, matching the test bed configuration.

5.2 Model Validation

There are no production implementations of PKCROSS or PKTAPP; so our approach, which is to begin by building "skeleton" implementations of these two protocols, is warranted. We developed the skeletons in standard C and used the RSA reference library, RSAREF [Johnson 2000], for public key encryption and Karn's DES library [Karn 2000] for symmetric key encryption. We implemented the KDCs and application servers on Microsoft Windows NT and the clients on Microsoft Windows 98. However, we did not use any operating-system-specific extensions in the C programs.

Figure 3 presents the software architecture of the skeletons. In the PKCROSS transaction, the client process steps through the standard Kerberos authentication message sequence to request service from an application server in a remote realm.

The primary purpose of the skeleton software on the client is to issue requests, quickly confirm the validity of the response, and timestamp the transaction to report response time statistics. The client-side processing has been

Table II. Encryption Operations for PKCROSS and PKTAPP with One Application Server

Authentication Transaction	No. Private Key Operations	No. Public Key Operations	No. Secret Key Operations
PKCROSS			
Client	0	0	7
Local KDC	2	3	5
Remote KDC	1	4	4
Application Server	0	0	3
Totals	3	7	19
PKTAPP			
Client	2	3	3
Application Server	1	4	4
Totals	3	7	7

simplified to focus on the shared resources: KDCs and application servers. The client process will loop through many transactions for the purpose of reporting average response time statistics.

A single process runs on the local KDC to accept client requests in UDP datagrams and use PKINIT to cross-authenticate the requests with the remote KDC. Two processes run on the remote KDC: one waits for standard Kerberos requests arriving as UDP datagrams, and the other opens a TCP listening socket and waits for PKINIT transactions. The architecture follows the PKINIT IETF draft recommendations that public key exchanges should use TCP to accommodate longer message sizes and reduce the potential for message fragmentation. All KDC and application server processes are multiple-threaded; when they receive a message, they dispatch a thread to process and respond to the request. In the final step of the transaction, the client authenticates to the application server using a ticket received from the remote KDC.

In the PKTAPP transaction, the client process has the same role and interacts with two server processes. It conducts a PKINIT exchange over a TCP connection to a multiple-threaded server process and obtains a service ticket. The client completes the authentication by sending a UDP datagram (a Kerberos Version 5 AP-REQ message) to a multiple-threaded process running on the same physical server.

The baseline PKCROSS and PKTAPP transactions are constructed with one application host in a remote realm. Because the environment under study is a large multiple-realm network, we assume that the client and KDCs must present certificates for authentication (i.e., no parties store certificate serial numbers and local copies of certificates). Further, we assume that the remote server must validate two certificates in a chain corresponding to the certificate signed by the local CA and a certificate signed by the remote CA. We apply the same assumptions for authenticating the local KDC to the remote KDC.

We configured the client, KDCs, and application server implementations to perform all encryption operations with 1024-bit RSA keys and standard DES. Table II summarizes the encryption operations performed. The total number of public and private key operations for authentication to a single application server is identical for PKCROSS and PKTAPP. This observation is intuitive because both protocols use the PKINIT message interchange for a single

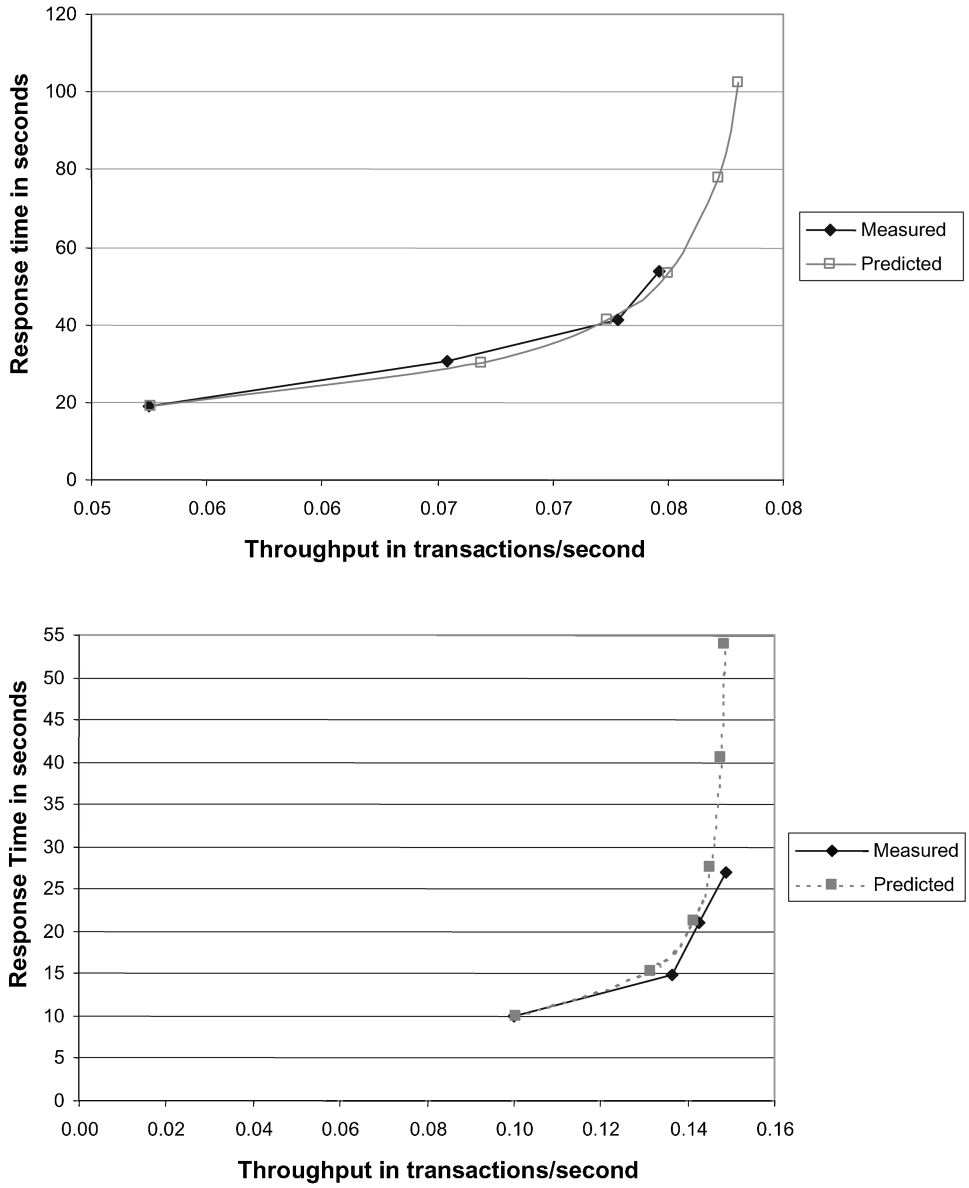

Fig. 4. Calibration results for the PKCROSS (top) and PKTAPP (bottom) transactions.

authentication between KDCs (in the case of PKCROSS) and between the client and the server (in the case of PKTAPP). PKCROSS requires more secret key operations because there are additional message exchanges among the KDCs.

The model's predictive accuracy for PKCROSS and PKTAPP transactions is shown in Figure 4. Both graphs demonstrate good calibration between the model and observed test bed results. The predicted response times and throughputs are within 3 percent of measured results.

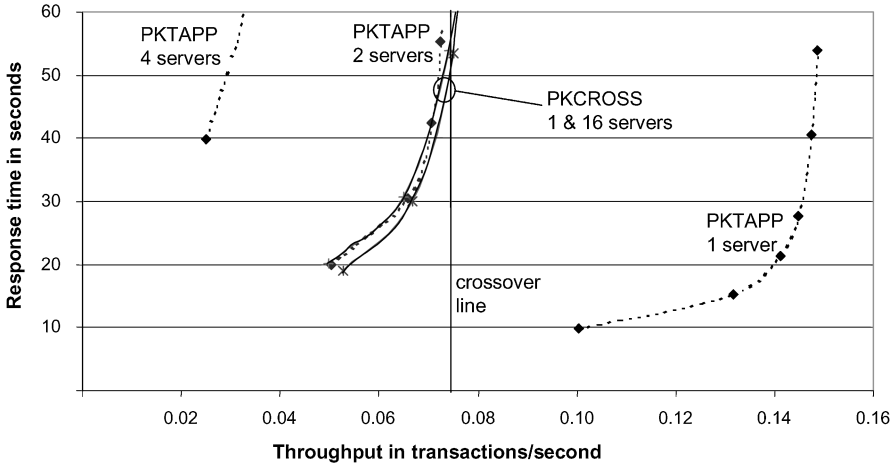

Fig. 5. Comparative PKCROSS and PKTAPP performance.

5.3 "What-If" Analysis

We used the validated model to investigate performance with an increased number of application servers. As the number of application servers increases, the number of "visits" made to the corresponding servers in each transaction increases. The PKTAPP transaction will include an additional set of public key authentication calculations for every additional application server. In the PKCROSS transaction, there is only one public key authentication—between the local and remote realm KDCs—regardless of the number of application servers in the remote realm.

Figure 5 presents the comparative performance—response time plotted as a function of PKCROSS and PKTAPP transaction throughput—for an increasing number of application servers. The transactions for PKCROSS represent authentication to one and sixteen application servers in the remote realm. The transactions for PKTAPP represent authentication to one, two, and four servers in the remote realm. We expect that uses similar to the example offered in at the beginning of this section could require authentication to four or more servers in a single realm.

The transaction rates for both protocols were increased until the overall response time became unstable and grew rapidly. In the PKCROSS transaction, the first bottleneck was the remote KDC processor, which had two processes running (one processing Kerberos UDP messages and the other listening for PKINIT transactions over TCP connections) and handled half the public key calculations for the KDC-to-KDC PKINIT exchange.

The next bottleneck, with very similar workload, was the local KDC. The application server was under-utilized; it conducted the final user authentication with only secret key encryption. Under PKCROSS, the KDCs remained the saturation point regardless of the number of remote realm application servers participating in the authentication.The indifference of the PKCROSS protocol to the number of servers in the remote realm is shown clearly in Figure 5.

The analysis demonstrates that, while PKTAPP is a significantly better performance choice for a single remote server, PKCROSS is significantly more stable for anything greater than two servers in the remote realm.

Our model accounts only for application server workload generated by authentications. In general, the application server will carry additional processing workload. If we had added that workload to the model, it would make PKCROSS look even more favorable.

We call the number of servers per realm that favor PKCROSS performance over PKTAPP the "crossover." The crossover may vary with server and network capacity. The test bed was constructed with low-performance servers and a 10-Mbps local area network, which has much better performance than typical Internet connections.

We repeated the model variations over a range of server and network performance. We increased the performance of KDCs and application servers (i.e., service times were reduced) by one and two orders of magnitude. As a result, we studied a range of processor performance that varied from a "1" to "100" SPEC CINT95 [SPEC 2000] rating—from a very-low-end Intel Pentium processor to a high-end server. Network performance ranged from LAN speeds to a network throughput of 12,750 bytes per second and a latency of 80.5 milliseconds to characterize slow Internet links [Menascé and Almeida 2002].

The results of our fast-processor/slow-network model are documented in Figure 6. This analysis indicates that as long as the capacity of the KDCs and applications servers are approximately the same, the performance benefits of PKCROSS, when more than two application servers are accessed in the remote realm, hold for increased processor capacity and reduced network throughput. One might first guess that the increased network time would favor PKTAPP because PKCROSS includes more message exchanges. The increased network delay does result in more separation between the PKCROSS response time curves for authentications of 1, 8, and 16 application servers. However, because PKTAPP sends large messages carrying certification chains to each application server, we observe the same crossover result.

The quantitative analysis was required to determine the contribution of network and processing delay in the response times of each protocol. The queuing model was required to project the shape of the response time curves. Figure 6 illustrates that with one or two application servers in the remote realm, the response time in the stable portion of the PKCROSS and PKTAPP curves (i.e., below 6 transactions per second) are close in value. However, as the response times become unstable (when the local KDC saturates), PKCROSS and PKTAPP result in significantly different performance profiles.

5.4 Summary of Proposed Protocol Enhancements and Benefits

We have demonstrated, through the use of validated analytical queuing models, the quantitative performance differences between two proposals to public-key-enable Kerberos, PKCROSS and PKTAPP. Our analysis shows that, over the range of server and network capacity studied, PKCROSS outperforms the simpler protocol, PKTAPP, for authenticating to more than one application

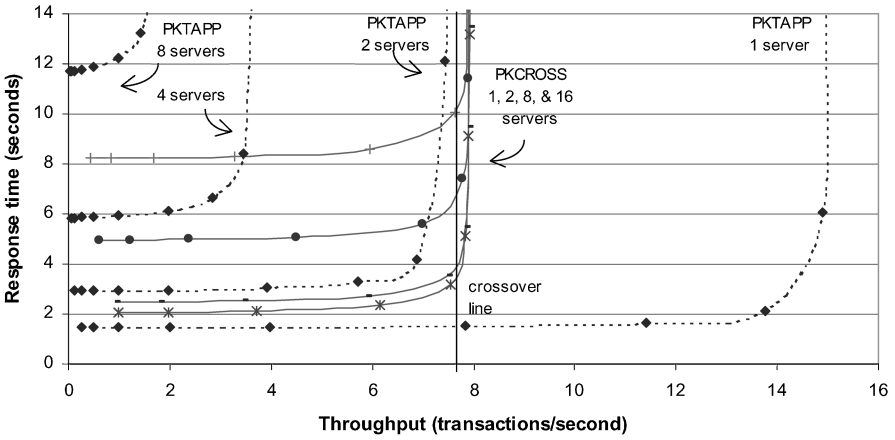

Fig. 6. Comparative PKCROSS and PKTAPP performance with increased server capacity and decreased network throughput.

server in a remote realm. This finding can be used to guide a high-level protocol that combines PKTAPP and PKCROSS to improve performance.

Use of such a high-level protocol would require that each application server provide support for both PKTAPP and traditional Kerberos. It would also require that the client know, a priori, the number of servers that would be authenticated to in a given realm. Neither requirement is onerous. The application server could support both PKTAPP and traditional Kerberos on two well-known ports. PKTAPP could be listening on a TCP socket; Kerberos could be awaiting UDP datagrams. The client, in the process of searching for information over a large number of servers, is often presented with a "hit list" before beginning the process of server authentications. This would allow the client to use either PKTAPP or PKCROSS based on the number of servers for each realm. This type of scenario fits well within the criminal investigation example offered at the beginning of this section.

6. EXAMPLE 2: THE ANALYSIS OF PUBLIC-KEY-ENABLED KERBEROS IN MOBILE COMPUTING ENVIRONMENTS

In a mobile computing context, the performance implications of protocol design are often accentuated by limitations in the capacity of the mobile processor and wireless network. The resources required to perform public key operations and transmit large messages may result in unacceptable performance characteristics and extended user authentication response times.

In this example, we explore the performance of user authentication from mobile devices. Specifically, we develop and analyze a variant of PKINIT that can use a proxy server to assist the mobile device. Adapting PKINIT to a mobile computing platform would provide a mature authentication mechanism and allow mobile users to operate in a PKI-supported environment.

Section 3 provided background information on current systems that can compensate for the resource limitations of mobile computing platforms by

off-loading processing to a proxy server (i.e., Charon and WAP WTLS). Questions remain about the performance value of proxy servers.

Proxies can help resource-constrained mobile devices communicate as peers on the Internet, but they can also introduce an additional layer of complexity, possibility of network delay, and opportunity for security breach. With the rapid increases in computing capacity of mobile devices such as PDAs and cell phones, it may not be beneficial to invest in proxy-based architectures and protocols that address what may be a short-term resource constraint.

We begin formulating our authentication protocol for the mobile platform by identifying a set of design guidelines. We have chosen guidelines that could lead to lower response times and address some of the limitations identified in Charon and WTLS:

Reduce the number of public/private key operations performed on the mobile platform. In general, public key operations consume significant processing resources and will adversely affect performance and user response time. In some algorithms, and depending upon the encryption parameters used, private key operations (e.g., signing) consume more compute resources than public key operations (e.g., signature verification). Minimizing the number of public and private key operations will always improve performance; trading a private key operation for a public key operation may improve performance.

When a proxy is used, maintain the option to preserve the encrypted data stream through the proxy. The fundamental criticism of WAP WTLS security is that it requires the data stream to be decrypted and re-encrypted in the proxy. PKINIT for mobile platforms should implement end-to-end security and should not require decryption in the proxy. However, if the proxy is proven to be trusted, it may be valuable to provide an option for the proxy to decrypt the data stream so that it can support the mobile platform in a manner similar to Charon.

Retain the standard Kerberos formats for messages sent to the KDC and application server. This will allow the protocol to be used with standard Kerberos KDCs and application server implementations.

Preserve the semantics of Kerberos. This will allow existing proofs of Kerberos authentication properties to be used in arguing that the new protocol achieves the same objectives.

Figure 7 maps the PKINIT protocol onto a mobile client, a KDC, and a target application server. We call this adaptation M-PKINIT. In M-PKINIT, only minor modifications are made to the PKINIT protocol. One modification is to use an optional feature, which appeared in a PKINIT draft that expired May 26, 1998, and has been removed in more recent drafts. This feature accommodates the operation of PKINIT if the client only possesses a signing key and can also be used with RSA, which allows both signing and encryption with the same key and algorithm.

In this situation, the client generates the session key and encrypts it with the KDC's public key. Normally, the KDC generates the session key and encrypts it with the client's public key. This feature swaps a private key operation for a public key operation on the mobile platform. It assumes that

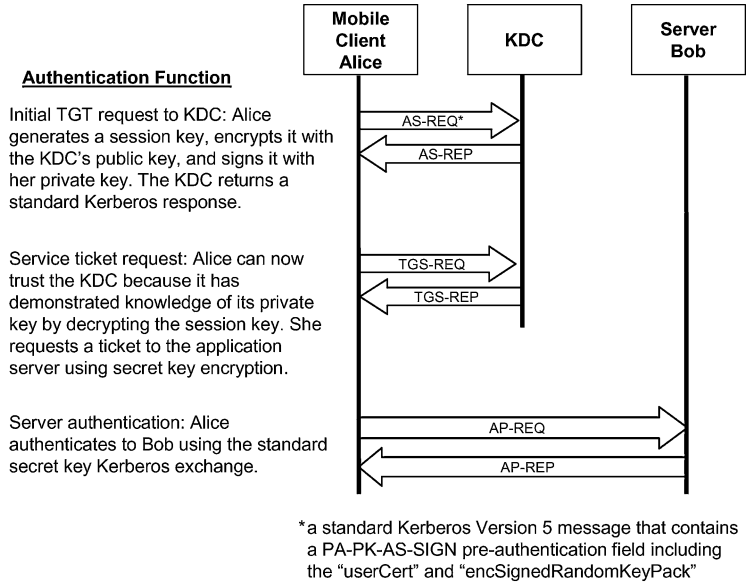

Fig. 7. M-PKINIT transaction.

the client knows the KDC's public key prior to receiving it as a part of the certification chain. One potential security risk is that the mobile client will not generate a session key that is strong enough. However, the KDC retains the option to reject the client-generated session key if it does not meet the KDC's policies for encryption strength.

Figure 8 maps the PKINIT protocol onto a mobile client, a KDC, a proxy, and an application server. We call this adaptation MP-PKINIT.

MP-PKINIT also saves a public key operation by allowing the client to generate the KDC session key. In the first message sent from the client to the proxy, the client has the option of revealing the KDC session key to the proxy. To do this, the client encrypts the session key with the proxy's public key so that the discovery of the session key requires knowledge of the proxy private key. The proxy introduces additional store-and-forward steps into the protocol, which will certainly increase processing and communications overhead. To mitigate this overhead, an additional shortcut is taken: the client's certificate chain is cached at the proxy, eliminating the need to transfer the client certificates over the wireless network.

6.1 QUEUING MODELING

The queuing model topology presented in Figure 1 also applies for M-PKINIT and MP-PKINIT if a mobile client is substituted for the client, a wireless network for the local area network, and a proxy server for the local KDC. We adjusted the branching probabilities of the modeled M-PKINIT and MP-PKINIT transactions to reflect the path they followed through the closed queuing network.

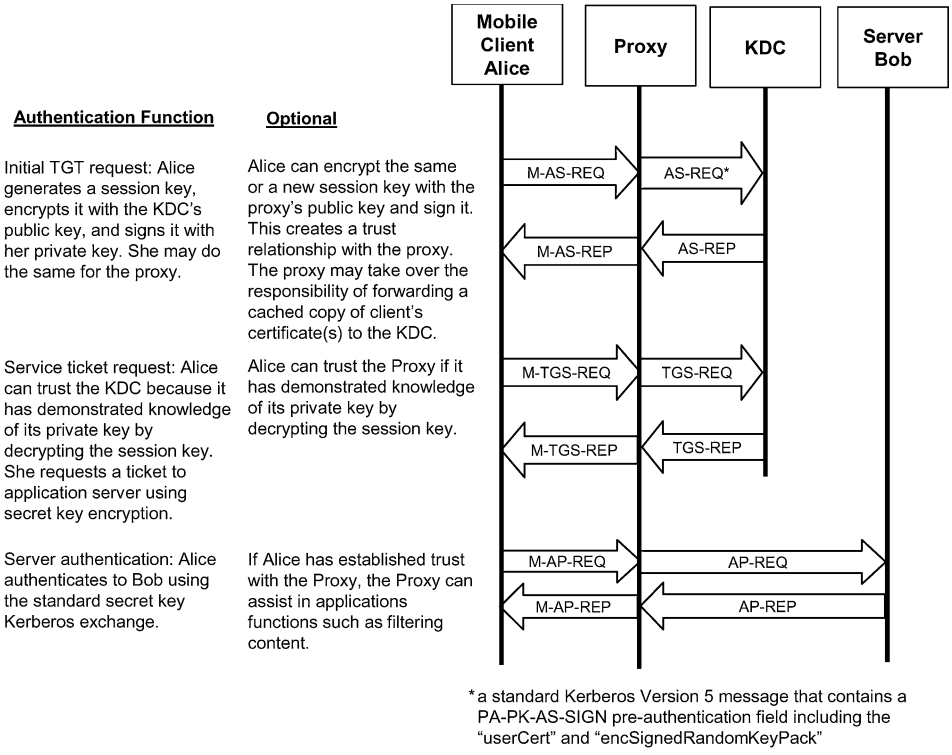

Fig. 8. MP-PKINIT transaction.

6.2 Modeling Validation

Lacking any implementation of the M-PKINIT and MP-PKINIT protocol designs to evaluate performance characteristics, we constructed implementations similar to the PKCROSS skeleton. In the new implementation, two processes run on the proxy and Kerberos KDC. One process, on both the proxy and the server, opens a TCP listening socket and waits for PKINIT transactions. The other process waits for standard Kerberos requests arriving as UDP datagrams. TCP connections are kept open as long as possible to reduce the effects of connection setup and "slow starts" [Stevens 1999]. For example, the proxy holds its connection with the client open while it communicates with the KDC on the client's behalf. All of the standard (i.e., non-public-key) Kerberos transactions use UDP. KDC, proxy, and application server processes are multiple-threaded. When they receive a message, they dispatch a thread to process and respond to the request. The mobile client either communicates entirely through the proxy or directly to the KDC and application server, depending on the protocol we are testing.

We focused on the mobile client device in the development and test configuration shown in Figure 3. The mobile device is a Vadem Clio C-1000, which uses Windows CE, a popular operating system for PDAs and other handheld devices. The C-1000 incorporates a 100-MHz MIPS R4000 CPU and has 16 MB of RAM available for programs and storage. The remaining processing elements—the

Table III. Service Time Measurements for M-PKINIT and MP-PKINIT

Protocol Variant	Time in Milliseconds for Protocol Phase				
	Pre-auth	Auth	Post-auth	TGT & ST	Total
1. M-PKINIT standard	864	3060	255	61	4240
2. M-PKINIT client session key	929	5404	4	65	6398
3. MP-PKINIT standard	866	3240	256	144	4507
4. MP-PKINIT client session key	920	5587	2	136	6646
5. MP-PKINIT trusted proxy	1773	8595	2	136	10506
6. MP-PKINIT proxy assist	1807	8421	3	142	10373

KDC, proxy, and servers—are configured as described in Example 1 (Section 5). The configuration includes conventional Windows client workstations to generate larger numbers of transactions and load the servers. We wrote the application in C++ and employed the same public and secret key encryption libraries as in Section 5.

The skeleton implementation and test platform allowed us to measure service time for an authentication transaction under a variety of protocol permutations. The protocol permutations selectively engage subsets of the features proposed in this section. Table III summarizes the results.

We recorded service times for six permutations:

(1) *M-PKINIT standard*: The unmodified PKINIT protocol skeleton running on the mobile client, a KDC, and an application server.
(2) *M-PKINIT client session key*: The PKINIT skeleton modified to allow the client to generate the session key and thereby trade a private key operation for a public key operation on the mobile platform.
(3) *MP-PKINIT standard*: The PKINIT skeleton modified to incorporate a proxy server. The proxy only performs store-and-forward operations.
(4) *MP-PKINIT client session key*: The PKINIT skeleton modified to incorporate a proxy server and allow the client to generate the session key.
(5) *MP-PKINIT trusted proxy*: The PKINIT skeleton modified to include an authentication between the client and the proxy and thereby establish a trust relationship.
(6) *MP-PKINIT proxy assist*: The PKINIT skeleton modified to include mutual authentication. In addition, the proxy caches and adds the client's certificate chain to the transaction as it passes through, reducing the message size across the wireless network.

Only in the *MP-PKINIT trusted proxy* and *MP-PKINIT proxy assist* transactions can the proxy provide added benefit to the client. In the other adaptations, the proxy acts as a pass-through node. The authentication transaction is divided into four segments for the purpose of service time measurement and analysis:

(1) *Pre-auth* includes the mobile device's processing prior to transmission of the first message.
(2) *Auth* includes transmission and Proxy/KDC processing time to process the first authentication message.

(3) *Post-auth* includes the mobile device's processing of the reply from the Proxy/KDC.

(4) *TGT & ST* includes all other client, Proxy, KDC, and application server processing related to the Kerberos Ticket Granting Ticket and Service Ticket.

Table III illustrates the impact when the client generates the session key. Row 2 service times reflect the increases in *Pre-auth* and *Auth* to account for the additional time required for the client to generate the key, sign it, and encrypt it with the KDC's public key, and then for the KDC to decrypt the key and verify the client's signature.

In contrast, there is a saving in PDA service time during the *Post-auth* step. Only the KDC can decrypt the session key with its private key, so the client can authenticate the KDC by confirming that the KDC properly encrypted the "EncKDCRepPart" portion of the message. No further public key operations are required in *Post-Auth* for the client to authenticate the KDC.

For the test configuration, the increase in KDC service time means that the M-PKINIT client session key transaction produces a longer response time than the M-PKINIT standard transaction. This effect is similar when the same feature is added to MP-PKINIT (rows 3 and 4 in Table III), although the total response time is higher because additional delays are introduced by communicating through the proxy. Even more delay is added when the proxy and client mutually authenticate to establish a trust relationship (rows 5 and 6 of Table III). The *MP-PKINIT proxy assist transaction* (row 6) does not produce a meaningful reduction in service time over the *MP-PKINIT trusted proxy* transaction (row 5) because the message size saving that results when the proxy caches the client certificate is not significant at the local area network speeds of the test configuration.

Figure 9 plots the results of the model's predictions against measured results for the test configuration. We recompiled and relinked the Windows CE client source code in a standard Win32 environment so that it could be run on a standard PC for the purpose of generating higher transaction rates and workload. Only very minor code changes were required for the port from CE to Win32. The figure demonstrates good calibration between the model and observed test bed results, supporting the model's predictive accuracy.

6.3 "What-If" Analysis

The performance of the components in a typical mobile computing environment may vary significantly from our test configuration. In particular, the network that connects all computers in the test configuration is a 10-Mbps Ethernet. Generally, data transfer rates in a wireless network will be significantly slower. The link between the KDC and the application server will most likely run at wide area network speeds rather than local area network speeds. On the other hand, the servers (i.e., KDC, proxy, and application servers) will most likely be more powerful than the low-end Pentium workstations we used for testing. We modified the transaction component service times in three ways

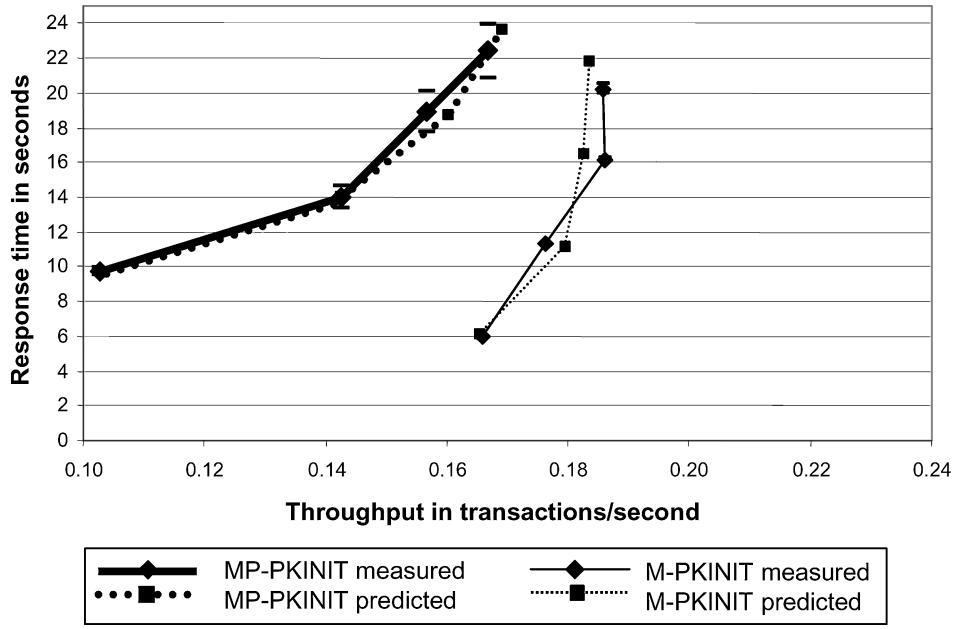

Fig. 9. Model calibration results.

for the "what-if" analyses:

(1) We lowered wide area network throughput to 127,500 bytes per second and added a latency of 44.5 milliseconds to reflect wide area Internet performance [Menascé and Almeida 2002].

(2) We used two wireless network transmission capacities—9600 bits per second with a latency of 520 milliseconds to represent 2G, and 384 Kbits per second with a latency of 100 milliseconds to represent 3G network capacities [PCIA 1998; Alanko et al. 1994; Liljebeg et al. 1996; Metricom 2001].

(3) We varied the server capacities by speed-up factors ranging from 30 to 100.

We expect at least a factor of ten improvement in server capacity over the low-powered PCs used—a comparison using benchmarks such as SPEC CINT suggests that multipliers as high as 100 are appropriate if comparing the CPU capacity of a high-end server to our test bed PCs.

Figure 10 presents an analysis of service time sensitivity to server and wireless network capacity. In this analysis, we used two wireless network capacities—at 2G and 3G levels—and we varied server capacities by a multiplier ranging from 1 to 20 times the capacity of those in the test bed. At both wireless network capacities we modeled, the *M-PKINIT client session key* protocol variant performed slightly better than the *M-PKINIT standard* variant as the server speed-up range hit the factor of ten.

A more interesting result is that adding the proxy service—in this case, caching certificates for the client—significantly reduces response times at 2G speeds from about 20 seconds to just above 14 seconds. This reduction is based on an average certificate size of 1.8 KB, consistent with the range of sizes of

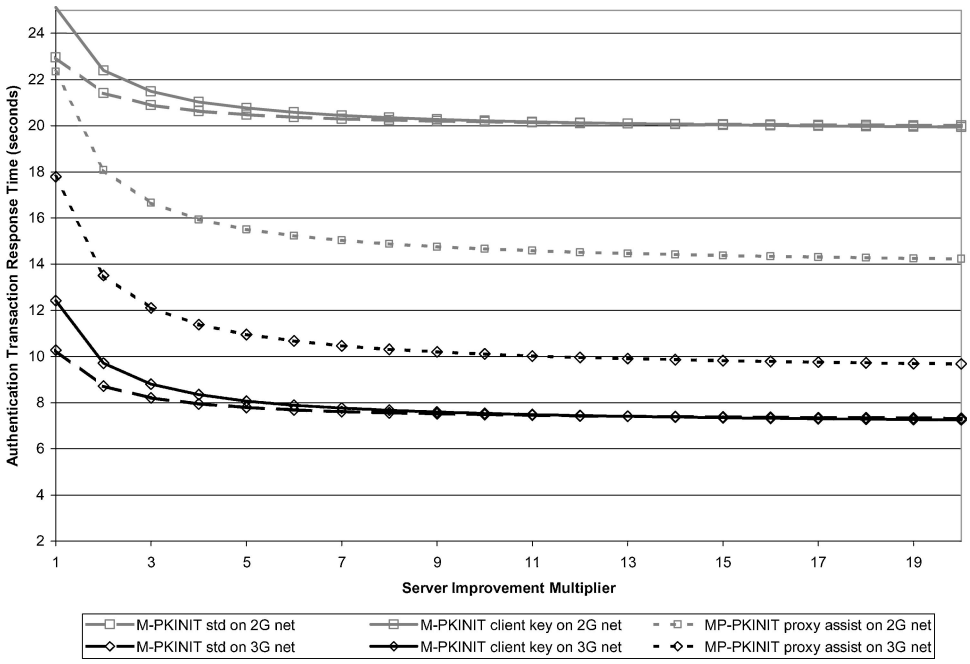

Fig. 10. Sensitivity to server and network capacity.

commercial certificates [Taschler 1997; OSD 2000]. When the wireless network throughput is increased to 3G speeds, the proxy is a response time burden and increases the response time by more than 2 seconds. The quantitative analysis was required to determine the relative response time contribution of processing and network delay leading to this observation.

Figure 11 presents the results of the queuing network model with a plot of M-PKINIT and MP-PKINIT authentication transaction throughput versus response time at a wireless network speed of 9600 bps and with several server speed-up multipliers. The figure shows a long, flat response time curve and a sharp knee for all modeled conditions. This is a result of the dominance of the wireless network delay in the total response time.

The wireless network was modeled as a fixed delay server—no increases in response time occurred as a result of increased authentication traffic. We made this assumption because we have no control over the amount of additional traffic going through the wireless network, and we would expect authentication traffic to be a negligible fraction of the overall traffic. We derived the throughput for the wireless network by degrading transmission speeds to account for frame errors using the measurements and analysis performed by Xylomenos and Polyzos [1999].

Figure 11 demonstrates that the response times start to climb rapidly at the point at which the KDC saturates and server delay exceeds wireless network delay. The KDC is the bottleneck server in all models. To underscore the role of the KDC, we decreased the acceleration of the proxy—the KDC was increased by a factor of 50, and the Proxy by a factor of 30. This made no noticeable change

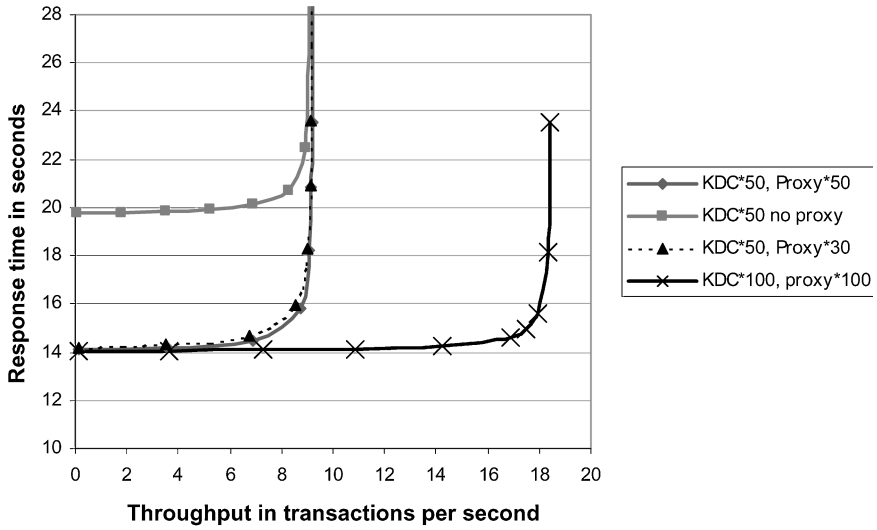

Fig. 11. Server-driven performance characteristics for M-PKINIT and MP-PKINIT.

in the response time curve, indicating that the proxy can be a lower capacity server than the KDC with little detrimental impact on user performance.

The nonproxy protocol curve knee occurs at the same place as the proxy version because the KDC workload is the same for proxy and nonproxy protocol variants. Finally, we observe the positive effect of increasing the capacity of the KDC by a factor of 100—more than double the achievable throughput.

The queuing network model analysis was required to confirm that our observations about the benefits of the proxy are invariant when the KDC and proxy servers are placed under load from multiple authentication users. If the proxy server saturated before the KDC, then it would be possible that, at some workload level, inclusion of the proxy in the 2G network would not result in improved performance. Figure 11 demonstrates that over a wide range of workload, the proxy has a positive effect on user authentication response time.

6.4 Summary of Proposed Protocol Enhancements and Benefits

At the beginning of this section, we defined four design guidelines for adapting Kerberos to a mobile computing environment. We can now assess the proposed PKINIT adaptations against these guidelines:

Did we reduce the number of public key operations performed on the mobile platform? By using an optional feature of PKINIT intended for situations in which only a signing key is available, we swapped a private key operation for a public key operation and eliminated the need for the client to validate the KDC's signature on the PKINIT reply message. The client did not have to verify the KDC's signature because the KDC can only decrypt the client-generated session key with the client's private key. If the KDC can reply with a message encrypted with the session key, it has effectively authenticated itself to the client. Both M-PKINIT and MP-PKINIT reduce the number of public key operations performed on the mobile platform.

When a proxy was used, did we maintain the options to preserve the encrypted data stream through the proxy? The client can choose whether or not to send the proxy a session key. The session key, encrypted with the proxy's public key, can be the same session key as that used for the KDC or it can be a unique key—it is the client's choice. MP-PKINIT provides the option for the client to preserve the encrypted data stream through the proxy.

Did we retain the standard Kerberos message formats to the KDC and application server? There are two difficulties in implementing MP-PKINIT and maintaining standard Kerberos message formats. First, Kerberos was not designed to communicate through a proxy, and several implementation details must be addressed to make this work. The IAKERB protocol specification works through these details. Second, the KDC must know and implement the PKINIT optional feature that allows the client to use a signing key. This feature is not in the current PKINIT draft. We observed that having the client generate the session key only nominally reduces overall response times, and then only if the KDC is not the bottleneck; it may not be worth changing current Kerberos implementations to support this feature. Beyond these two areas, both M-PKINIT and MP-PKINIT can employ standard Kerberos message formats at the interfaces to the KDC and application server.

Did we preserve the semantics of Kerberos? The introduction of the proxy and the client-generated session key represents a change to the semantics of Kerberos. The change is significant enough to require reformulation of the Kerberos formal logic arguments before one could assert that M-PKINIT and MP-PKINIT have the same authentication properties as Kerberos.

Our measurements and models have demonstrated that public key Kerberos is a feasible candidate for authentication in a mobile environment. We achieved reasonable performance with a well-proven public key encryption algorithm: RSA with 1024-bit keys. At G2 wireless network speeds (9600 bps), assistance is required from a proxy server in order to reduce total authentication response times—the proxy reduces the wireless network message traffic by caching certificates for the client. The current IETF draft for PKINIT allows the KDC to store client private keys. While this would eliminate the need to transmit certificates, it would also limit the protocol's use to situations in which the client was preregistered with the KDC.

7. SUMMARY AND FUTURE WORK

Performance is an important design consideration in authentication protocols. Our primary contribution is the development of a methodology for analyzing the performance characteristics of alternatives in authentication protocol design. This methodology has the following benefits and features:

The predictive step is based on closed queuing models. We have selected a technique that combines a class-switching formulation with an efficient solution method. This combination of techniques allows us to model some of the unique features of authentication protocols. For example, authentication protocols

often include multiple encryption steps that alternately apply asymmetric and symmetric algorithms. Consecutive requests for service to processing or communications resources may have widely variant service demands. The resultant service time distributions are difficult to model analytically. The class-switching formulation provides an approach to capture the performance characteristics of this type of resource consumption profile within the constraints required to compute an efficient numerical solution.

We use "skeleton implementations" to validate the predictive model. If a designer seeks to evaluate new features, operational protocol implementations may not be available and it may be difficult to validate the model against empirical measurements. In our methodology, we develop software skeletons that capture the performance characteristics of the protocol without requiring a full implementation.

The general predictive methodology can be used to assess performance under a widely variant set of operating conditions. In contrast, experimental testing can be limited in its ability to support performance evaluation under the range of operating conditions that the protocol under study may be subjected to in production use. The closed queuing network models can accommodate a wide range of input parameters and be used to explore performance under many different applications of the studied protocol.

In developing the examples to illustrate our methodology, we have made several contributions in the area of authentication protocol design based on Kerberos. In our first example, we analyze two proposed, public key variants of Kerberos: PKCROSS and PKTAPP. Our analysis shows that, over the range of server and network capacity studied, PKCROSS outperforms the simpler protocol, PKTAPP, for authenticating to more than one application server in a remote realm. This finding can be used to guide a high-level protocol that combines both PKTAPP and PKCROSS to improve performance.

In our second example, we demonstrate that a proxy server can be used between the client and the server to significantly improve public key-enabled Kerberos performance in a mobile setting. We show that as wireless network speeds increase, the role of the proxy is less beneficial from a performance perspective and may negatively impact response time.

There is more work to be done to further develop the two examples and to refine the methodology. The development of these protocols would undoubtedly benefit from extending the what-if analyses to a broader range of underlying encryption protocols and potential operational environments. Further, we selected our examples from proposed variants of the Kerberos authentication protocol. These protocols have similar, but not identical security semantics. Our methodology most directly applies to the analysis of protocols with identical security semantics, but differing performance characteristics. A more general problem would be to extend the methodology to compare protocols with differing security and performance characteristics so as to understand security-performance trade-offs. We believe that our analysis methodology is applicable to a broad range of protocols. Experimentation with analysis of a wider variety of authentication and security protocol designs and an extension to security/performance trade-offs remains as future work.

APPENDIX A. PROTOCOL SPECIFICATIONS FOR PKCROSS AND PKTAPP

The essential elements (i.e., excluding addressing and policy information) of the PKCROSS protocol are specified below using the following notation:

A is the client's name (Alice);

B is the applications server's name (Bob);

KDC-L and KDC-R are the local and remote Key Distribution Centers; $realm_L$ and $realm_R$are the names of the local and remote realms;

K_x is principal X's long-term secret key;

K_{xy} is a session key used to communicate between principals X and Y;

K_S is the remote KDC's secret key used for cross-realm authentications;

K_{temp} is a temporary key used in the PKINIT exchange between local and remote KDCs;

K_{random} is a key used to encrypt the PKINIT reply;

$PRIV_X$ is principal X's private key;

PUB_X is principal X's public key;

N is a nonce;

T is a timestamp; and

CC_X is the certificate chain for principal X.

Message 1. A → KDC-L: A, "krbtgt," $realm_L$, N

Message 2. KDC-L → A: $realm_L$, A, {"krbtgt," K_{aKDC-L}, A, T}K_{KDC-L}, {K_{aKDC-L}, N, T, "krbtgt"}K_a

Message 3. A → KDC-L: {"krbtgt," K_{aKDC-L}, A, T}K_{KDC-L}, {A, $realm_L$, T, N}K_{aKDC-L}, "krbtgt", $realm_R$, N

Message 4. KDC-L → KDC-R: T, N, {T, N}$PRIV_{KDC-L}$, CC_{KDC-L}, KDC-L, "pkcross," N

Message 5. KDC-R → KDC-L: {K_{temp}}PUB_{KDC-L}, {K_{random}, N, {K_{random}, N}$PRIV_{KDC-R}$}K_{temp}, CC_{KDC-R}, KDC-L, {"pkcross", $K_{KDC-L,KDC-R}$, KDC-L, T}K_S, {$K_{KDC-L,KDC-R}$, N, T, "pkcross"}K_{random}

Message 6. KDC-L → A: $realm_L$, A, {"pkcross," K_{aKDC-R}, A, T}$K_{KDC-L,KDC-R}$, {"pkcross", $K_{KDC-L,KDC-R}$, KDC-L, T}K_S, {$K_{a,KDC-R}$, N, T, "krbtgt"}$K_{a,KDC-L}$

Message 7. A → KDC-R: {"krbtgt," $K_{a,KDC-R}$, T}$K_{KDC-L,KDC-R}$, {"pkcross", $K_{KDC-L,KDC-R}$, KDC-L, T}K_S, {A, $realm_L$, T, N}K_{aKDC-R}, B, $realm_R$, N

Message 8. KDC-R → A: A, $realm_L$, B, $realm_R$, {B, K_{ab}, T}K_b, {K_{ab}, N, T, B, $realm_R$}$K_{a,KDC-R}$

Message 9. A → B: {B, K_{ab}, T}K_b, {A, $realm_L$, T, N}K_{ab}

Message 10. B → A: {T, N}K_{ab}

This specification assumes that the RSA algorithm is used so that digital signatures are accomplished by encrypting a message with a private key. In the first message, Alice requests a ticket granting ticket (TGT) from her local KDC. Her local KDC replies with a TGT and a session key encrypted with Alice's long-term secret key. In Message 3, Alice requests a TGT for the remote realm. She includes her local TGT and an authenticator. The authenticator contains an indication of her identity and a timestamp, both encrypted with the session

key. At this point, the local KDC must establish a trust relationship with the remote KDC using PKCROSS. That is accomplished in Messages 4 and 5. In Message 4, the local KDC sends the first PKINIT message to the remote KDC. This message includes an authenticator signed by the local KDC. In Message 5, the remote KDC responds as per PKINIT with a TGT. The remote KDC uses a special secret key for cross-realm authentications (K_S) for encrypting the cross-realm ticket. The remaining transactions use secret key encryption and standard Kerberos Version 5. In Message 6, the local KDC passes the remote TGT back to Alice. The local KDC includes the cross realm ticket as well as a ticket to the remote KDC. Alice requests a service ticket to application server Bob in Message 7. She must include the cross realm ticket in this request so that the remote KDC can extract the session keys. In Message 8, she receives her ticket to Bob. Finally, Alice authenticates to Bob in Message 9 and 10. If there are multiple application servers in the remote realm, Alice repeats Messages 7 through 10 to authenticate to each server.

The essential elements of the PKTAPP protocol are specified below, using the same notation conventions followed to specify PKCROSS.

Message 1. $A \rightarrow B$: T, N, {T, N}$PRIV_A$, CC_A, A, B, N
Message 2. $B \rightarrow A$: {K_{temp}}PUB_a, {K_{random}, N, {K_{random}, N}$PRIV_b$}K_{temp}, CC_b, A, {B, K_{ab}, A, T}K_b, {K_{ab}, N, T, B}K_{random}
Message 3. $A \rightarrow B$: {B, K_{ab}, T}K_b, {A, $realm_L$, T, N}K_{ab}
Message 4. $B \rightarrow A$: {T, N}K_{ab}

In Message 1, Alice sends her signed authenticator and her certificate chain in the request to obtain a service ticket to Bob. Bob responds with an encrypted, signed random key and a service ticket with an encrypted session key in Message 2. Alice now has a service ticket, but she still must authenticate to server Bob. This final authentication is accomplished with Message 3 and 4 using the secret session key that Bob associated with the service ticket. If Alice wishes to authenticate to other application servers in the remote realm, she must do so by repeating Messages 1 through 4 with each server.

APPENDIX B. THE MULTI-CLASS/CLASS-SWITCHING QUEUING NETWORK SOLUTION

The Mean Value Analysis (MVA) algorithm provides a stable, exact, iterative solution for closed queuing networks. It is based on three relations that hold for product form queuing networks models: the arrival theorem, the forced flow law, and Little's formula. The arrival theorem states that a new customer arriving at a queuing server will join a line that has the same average length as the total average length of the same queuing station in a network with one less customer. The forced flow law relates the system throughput (λ_0) with the throughput for individual queuing servers (λ_i). Little's formula states that average number of customers in line at a queuing servers is equal to the arrival rate to the server times the average wait for service.

These theorems can be combined to derive the equations behind the iterative MVA method. Before writing these equations, we note that our queuing network

solution must take into account situations in which different customers waiting at a queuing server have different mean service times and different branching characteristics upon departure from a station. To model these situations, requires the introduction of classes of customers and the ability of customers to switch from one class to another. Bruell and Balbo [1980] have generalized the MVA algorithm to allow for these features. Bruell and Balbo first group the classes of users into *equivalence classes* (ECs). An EC is a closed group such that there is zero probability the each customer within a specific EC will switch to class outside of the EC. Further, all classes within an EC are reachable from each other. The MVA equations can then be written as:

$$W_{iq}(\vec{n}_U) = s_{iq}(1 + L_i(\vec{n}_U - \vec{1}_q)) \tag{B.1}$$

$$\lambda_{0q}(\vec{n}_U) = \frac{n_q}{\sum_{i=1}^{M} v_{iq} W_{iq}(\vec{n}_U)} \tag{B.2}$$

$$\lambda_{iq}(\vec{n}_U) = \lambda_{0q}(\vec{n}_U) v_{iq} \tag{B.3}$$

$$L_{iq}(\vec{n}_U) = \lambda_{iq}(\vec{n}_U) W_{iq}(\vec{n}_U) \tag{B.4}$$

$$L_i(\vec{n}_U) = \sum_{q=1}^{U} L_{iq}(\vec{n}_U) \tag{B.5}$$

for U ECs numbered from $q = 1, 2, \ldots U$ where:

$\vec{n}_U =$ the vector $(n_1, \ldots, n_q, \ldots, n_U)$ of the number of customers in each EC for the current system state;

$W_{iq}(\vec{n}_U) =$ the mean response time at server i $(i = 1, \ldots, M)$ for EC q with $\vec{n}_U$ customers in the system;

$s_{iq} =$ the average service time for an EC q customer at server $i (i = 1, \ldots, M)$;

$L_{iq}(\vec{n}_U) =$ the queue length at server i $(i = 1, \ldots, M)$ when there are $\vec{n}_U$ customers in the system;

$v_{iq} =$ the visit ratio of customers in EC q at server $i (i = 1, \ldots, M)$;

$\lambda_{iq} =$ the throughput of customers in EC q at server $i (i = 1, \ldots, M)$;

$\vec{1}_q =$ the vector $(0, \ldots, 1, \ldots, 0)$ where all components except for the qth, which is equal to 1, are zero.

Equation (B.1) states the arrival theorem. Equation (B.2) is Little's Law for the entire system. Equation (B.3) is the forced flow theorem. Equation (B.4) is Little's Law for each queuing station. Finally, Eq. (B.5) states that the total queue length at server i is equal to the sum of the queue lengths at server i due to customers from all ECs. Note that Eq. (B.1) depends upon a queue length with a customer population that is one less than the current state of the system. As a result, MVA is an iterative algorithm. The solution for the full system is computed by iterating equations (B.1) through (B.5) for all M devices in the system, starting from $\vec{n}_U = \vec{0}$ up to the full number of customers in each EC.

The visit ratios, v_{iq}, in Eqs. (B.2) and (B.3) can be calculated from the visit ratios for the classes r of the original problem as follows:

$$v_{iq} = \frac{\sum_{r \in EC_q} v_{ir}}{\sum_{r \in EC_q} v_{0r}}. \tag{B.6}$$

The visit ratios to each server in each class (v_{ir}) are determined by the switching properties of each customer of class r in equivalence class EC_q.

Equations (B.1) through (B.5) provide performance metrics based on equivalence classes. Bruell and Balbo [1980] provide the following equations to convert from the equivalence classes q back to the classes r:

$$\alpha_{ir} = \frac{v_{ir}}{\sum_{s \in EC_q} v_{is}} \tag{B.7}$$

$$W_{ir}(\vec{n}_U) = s_{ir}(1 + L_i(\vec{n}_U - \vec{1}_q)) \tag{B.8}$$

$$\lambda_{ir}(\vec{n}_U) = \alpha_{ir}\lambda_{iq}(\vec{n}_U) \tag{B.9}$$

$$n_{ir}(\vec{n}_U) = \lambda_{ir}(\vec{n}_U) W_{ir}(\vec{n}_U) \tag{B.10}$$

The validity and accuracy of these equations require some restrictions on the server queuing disciplines and service-time distributions. If we want different service times for each class (as one would expect with different types of encryption performed at different phases of the protocol) we are limited to either "processor sharing" or "infinite server" queuing disciplines. In a processor sharing queue, service begins immediately for each customer joining the service center, but the service time lengthens in proportion to the total number of customers currently being serviced. This is a good model for multitasking, time slice scheduling operating systems. The customer at an infinite server experiences fixed delays independent of the number of customers at the service center. This type of server is appropriate to modeling the public networks in our problem. We expect that a very large user community will share the network and that workload increases from just our authentication users will not substantially impact the network response time.

For large numbers of customers, the MVA algorithm results in a large number of iterations. In order to eliminate iterations, we use a fixed-point approximation (AMVA) developed by Schweitzer and Bard [Schweitzer 1991]. Specifically, we rewrite Eq. (B.1) as follows:

$$W_{iq}\left(\vec{n}_U\right) = s_{iq}\left[1 + \frac{n_q - 1}{n_q} L_{iq}(\vec{n}_U) + \sum_{s=1 \& s \neq q}^{U} L_{is}(\vec{n}_U)\right], \tag{B.11}$$

where n_q is the total number of customers in equivalence class q.

With Eq. (B.11) substituting for Eq. (B.1), we can implement a multiclass/class-switching AMVA algorithm by successively solving Eqs (B.1) through (B.5) until the error in subsequent L_{iq}'s is less than some threshold value ε.

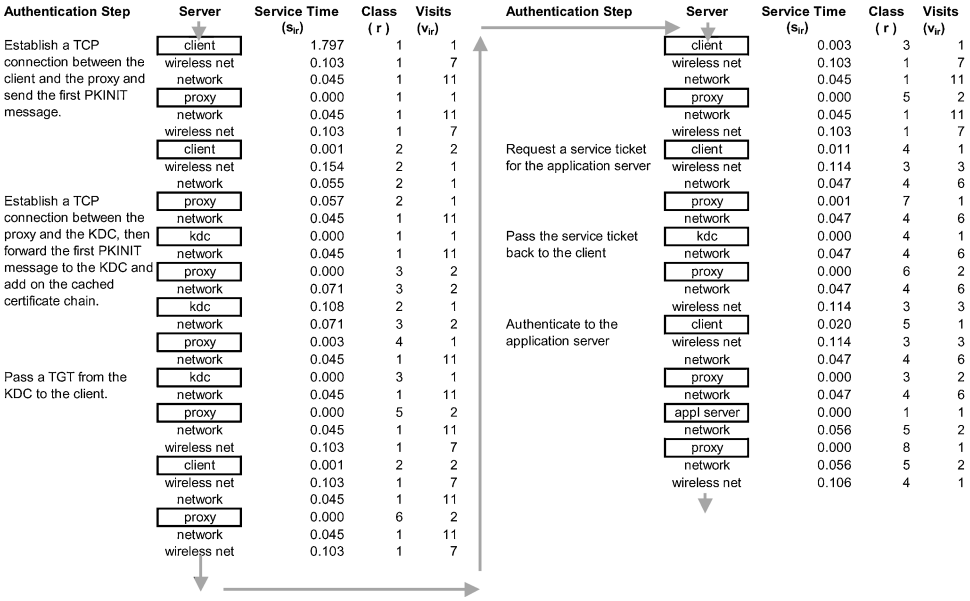

Fig. B.1. The MP-PKINIT transaction and its modeling parameters.

To apply these equations, we must express the authentication transaction in terms of the inputs to the AMVA algorithm: s_{ir} and v_{ir}. Figure B.1 illustrates how a sample transaction is mapped onto the AMVA inputs. We measured the service time and visit counts in Figure B.1 from the test bed described in Section 3 of this article. The numbers in Figure B.1 have been adjusted to represent a server speed-up factor of 50 (over our test bed servers) and 3G wireless networks. We conduct further what-if modeling by modifying the visit counts, service times, and the number of customers to reflect the range of operational conditions.

ACKNOWLEDGMENTS

The authors would like to extend special thanks to the reviewers for comments that greatly improved the quality of the final article.

REFERENCES

ALANKO, T., KOJO, M., LAAMANEN, H., LILJEBERG, M., MOILANEN, M., AND RAATIKAINEN, K. 1994. Measured performance of data transmission over cellular telephone networks. Department of Computer Science, Report C-1994,-53. University of Helsinki, Helsinki, Finland. November.

APOSTOLOPOULOS, G., PERIS, V., AND SAHA, D. 1999. Transport layer security: How much does it really cost? In *Proceedings of IEEE INFOCOM'99* (New York, New York, March). IEEE Computer Society Press, Los Alamitos, Calif., pp. 717–725.

ASHELY, P., AND BROOM, B. 1997. A survey of secure multi-domain distributed architectures. Faculty of Information Technology, Queensland University of Technology, Queensland, Australia.

BASSHAM, L. E. 1999. Efficiency testing of ANSI C implementations of round 1 candidate algorithms for the advanced encryption standard. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

BLAZE, M. 1996. *High-Bandwidth Encryption with Low-Bandwidth Smartcards*. D. Gollman, ed. Lecture Notes in Computer Science, vol. 1039. Springer-Verlag, Cambridge, UK, 33–40.

BRUELL, S. C. AND BALBO, G. 1980. *Computational Algorithms for Closed Queueing Networks*. Elsevier North Holland, Inc., New York, NY.

BURROWS, M., ABADI, M., AND NEEDHAM, R. 1990. A logic of authentication. *ACM Trans. Comput. Syst. 8*, 1, 18–36.

CRISTIAN, F. 1995. Exception handling and tolerance of software faults. In *Software Fault Tolerance*, M. R. LYU, ed. Wiley, Chichester, UK, 81–107.

CYLINK 2000. Closing the "Gap in WAP." www.securitytechnet.com, http://www.securitytechnet.com/resource/rsc-center/vendor-wp/cylink/Gapinwap.pdf.

DAI, W. 1999. Crypto++ 3.1 benchmarks. www.eskimo.com, http://www.eskimo.com/~weidai/benchmarks.html.

EL-HADIDI, M. T., HEGAZI, N. H., AND ASLAN, H. K. 1999. Performance evaluation of a new hybrid encryption protocol for authentication and key distribution. In *Proceedings of the 4th IEEE International Symposium on Computers and Communications* (Red Sea, Egypt, July). IEEE Computer Society Press, Los Alamitos, Calif.

FOX, A., AND GRIBBLE, S. D. 1996. Security on the move: Indirect authentication using Kerberos. In *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking* (Rye, New York, Nov.). ACM, New York, 155–163.

GROSS, D., AND HARRIS, C. M. 1998. *Fundamentals of Queueing Theory*. Third ed. Wiley, New York.

HARBITTER, A., AND MENASCÉ, D. A. 2001a. Performance of public key-enabled Kerberos authentication in large networks. In *Proceedings of 2001 IEEE Symposium on Security and Privacy* (Oakland, Calif., May). IEEE Computer Society, Los Alamitos, Calif., 170–183.

HARBITTER, A., AND MENASCÉ, D. A. 2001b. The performance of public key-enabled Kerberos authentication in mobile computing applications. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8)* (Philadelphia, Penn., Nov.). ACM New York, 78–85.

JOHNSON, M. 2000. North American cryptography archives. www.cryptography.org, http://cryptography.org/ cgi-bin/noexport.cgi.

JORMALAINEN, S., AND LAINE, J. 1999. Security in the WTLS. Helsinki University of Technology, Helsinki, Finland.

KARN, P. 2000. The crypto CD. www.cryptocd.org, ftp://ftp.pini.org/pub/cryptocd/source/cyphers/des/c/karn/.

KHARE, R. 1999. W* effect considered harmful, 4K Associates. *IEEE Internet Comput* (July-Aug.), 89–92.

LAMBERT, P. 1998. Elliptic curve cryptography delivers high performance and security for e-commerce. *Comput. Sec. J. XIV*, 4, 23–29.

LILJEBEG, M., HELIN, H., KOJO, M., AND RAATIKAINEN, K. 1996. Mowgli WWW software: Improved usability of WWW in mobile WAN environments. In *Proceedings of IEEE GLOBECOM '96* (Westminster, London, England, Nov.), IEEE Computer Society Press, Los Alamitos, Calif., 33–37.

MARTINKA, J. J., FRIEDRICH, R. J., FRIEDENBACH, P. M., AND SIENKNECHT, T. F. 1993. A performance study of DCE 1.0.1 cell directory service: Implications for application and tool programmers. *Networked Systems Architecture, Hewlett-Packard*. International Workshop OSF DCE, Karlsruhe, Germany, Oct.

MEDVINSKY, A., HUR, M., AND NEUMAN, C. 1997. Public key utilizing tickets for application servers (PKTAPP). www.ietf.org, http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-tapp-03.txt.

MENASCÉ, D. A. AND ALMEIDA, V. A. F. 2000. *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice-Hall, Upper Saddle River, N.J.

MENASCÉ, D. A. AND ALMEIDA, V. A. F. 2002. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice-Hall, Upper Saddle River, N.J.

METRICOM 2001. *Ricochet Security Whitepaper*, Apr. Metricom, Inc.

NEEDHAM, R. M. AND SCHROEDER, M. D. 1978. Using encryption for authentication in large networks of computers. *Commun. ACM 21*, 993–999.

NEUMAN, B. AND TS'O, T. 1994. Kerberos: An authentication service for computer networks. *IEEE Commun. Magazine 32*, 9, 33–38.

OPENGROUP 1997. DCE 1.1: Authentication and Security Services. www.opengroup.org, http://www.opengroup.org/publications/catalog/c311.htm.

OROZCO-BARBOSA, L., SERRANO, L., AND QUIROZ, E. 1998. Performance evaluation of the IS-41 security mechanisms in a PCS supporting intelligent network services. In *Proceedings of IEEE IN'98 Workshop* (Bordeaux, France, May). IEEE Computer Society Press, Los Alamitos, Calif., 142–154.

OSD 2000. Consideration of smart cards as the DoD PKI authentication device carrier, Office of the Secretary of Defense. www.c3i.osd.mil, http://www.c3i.osd.mil/ebpublic/smartcardreport.pdf.

PERSONAL COMMUNICATIONS INDUSTRY ASSOCIATION. 1998. Market demand forecast for terrestrial third generation (IMT-2000) service for the Personal Communications Industry Association. www.pcia.com, www.pcia.com/advocacy/3gstudy.htm.

SCHWEITZER, P. J. 1991. A survey of mean value analysis, its generalizations, and applications, for networks of queues. William I. Simon Graduate School of Business Administration, University of Rochester, Rochester, N.Y.

SIRBU, M. A., AND CHUANG, J. C.-I. 1997. Distributed authentication in Kerberos using public key cryptography. In *Proceedings of Symposium on Network and Distributed System Security* (San Diego, Calif., Feb.), IEEE Computer Society Press, Los Alamitos, Calif.

SPEC 2000. CPU95 benchmarks. http://www.spec.org/osg/cpu95/.

STALLINGS, W. 1994. Kerberos keeps the enterprise secure. In *Data Communications*, 103–111.

STEVENS, W. R. 1999. *TCP/IP Illustrated*, vol. 1. Addison-Wesley, Reading, Mass.

SWIFT, M., TROSTLE, J., ABOBA, B., AND ZORN, G. 2001. Initial and padd through authentication using Kerberos V5 and the GSS-API (IAKERB), IETF. www.ietf.org; http://search.ietf.org/internet-drafts/draft-ietf-cat-iakerb-06.txt.

TASCHLER, S. 1997. *Datakey CIP 3.0 Whitepaper*. www.datakey.com; http://www.datakey.com/cardpage/cip3_whitepaper.htm.

TUNG, B., RYUTOV, T., NEUMAN, C., TSUDIK, G., SOMMERFIELD, W., MEDVINSKY, A., AND HUR, M. 1998. Public key cryptography for cross-realm authentication in Kerberos. www.internic.net; http://www.internic.net/internet-drafts/draft-ietf-cat-derberos-pk-cross-03.txt.

TUNG, B., NEUMAN, C., HUR, M., MEDVINSKY, A., MEDVINSKY, S., WRAY, J., AND TROSTLE, J. 2001. Public key cryptography for initial authentication in Kerberos. www.ietf.org, http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt.

WAP 2000a. Wireless application protocol TLS profile and tunneling specification. www.wapforum.org, http://www1.wapforum.org/tech/documents/WAP-219-TLS-20010411-a.pdf.

WAP 2000b. Wireless application protocol wireless transport layer security specification, wireless application forum, Ltd. 2000. www.wapforum.org, http://www1.wapforum.org/tech/documents/WAP-261-WTLS-20010406-a.pdf.

XYLOMENOS, G., AND POLYZOS, G. C. 1999. Internet protocol performance over networks with wireless links. *IEEE Netw. 13*, 4, 55–63.

ZENEL, B. 1999. A general purpose proxy filtering mechanism applied to the mobile environment. *Wireless Netw. 5*, 391–409.

ZORKADIS, V. 1994. Security versus performance requirements in data communications systems. D. Gollman, ed. In *Proceedings of Computer Security—ESORICS 94* (Brighton, U.K., Nov.) 19–30.

Received July 2001; revised January 2002 and June 2002; accepted June 2002