The Performance of Public Key-Enabled Kerberos Authentication in Mobile Computing Applications

Alan Harbitter
PEC Solutions, Inc.
12750 Fair Lakes Circle
Fairfax, VA 22033
703-679-4900
aharbitter@pec.com

Daniel A. Menascé
Department of Computer Science
George Mason University
Fairfax, VA 22030
703-993-1530
menasce@cs.gmu.edu

ABSTRACT
Authenticating mobile computing users can require a significant amount of processing and communications resources—particularly when protocols based on public key encryption are invoked. These resource requirements can result in unacceptable response times for the user. In this paper, we analyze adaptations of the public key-enabled Kerberos network authentication protocol to a mobile platform by measuring the service time of a "skeleton" implementation and constructing a closed queuing network model. Our adaptation of Kerberos introduces a proxy server between the client and the server to mitigate potential performance deficiencies and add functional benefits. Our analysis indicates that assistance from the proxy makes public key Kerberos a viable authentication protocol from a performance perspective. However, as wireless network speeds increase from current 2G levels to the 3G targets, the proxy can become a response time liability. The proxy's role in the protocol, while warranted in current applications, will have to be re-modeled and re-considered as both wireless transmission speeds and proxy processing speeds increase.

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
Measurement, Performance, Design, Security.

Keywords
Mobile computing, Authentication, Performance modeling, Proxy servers, Kerberos, Public key cryptography.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS'01, November 5-8, 2001, Philadelphia, Pennsylvania, USA.

Copyright 2001 ACM 1-58113-385-5/01/00011…$5.00.

1. INTRODUCTION
Network authentication protocols regularly employ public key cryptography to securely identify clients to servers and establish trust relationships. In a mobile computing context, the processing and communications resources required to perform public key operations and transmit large messages may result in unacceptable performance characteristics and extended user authentication response times. Several current systems compensate for the resource limitations of mobile computing platforms by off-loading processing to a proxy server. The Wireless Application Protocol (WAP) gateway is an example of widespread use of proxies in mobile computing user authentication. The WAP gateway converts between the reduced function WAP authentication and standard Internet protocols. There is some question about the value of proxy servers. Proxies can perform a valuable service by helping resource-constrained mobile devices communicate as peers on the Internet. However, they can introduce an additional layer of complexity, network delay, and opportunity for security breach. With the rapid increases in computing capacity on personal digital assistants (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.

This paper explores the performance of user authentication from mobile devices. Specifically, we develop and analyze a public key-based variant of the Kerberos [1] authentication protocol that can employ a proxy server to assist the mobile device. Section 2 provides background on related proxy-supported mobile authentication systems that have been proposed or are in use. In Section 3, we describe our proposal for a mobile, public key-enabled Kerberos protocol called M-PKINIT (when used without a proxy) and MP-PKINIT (when used with a proxy). We also describe a "skeleton implementation" of the protocol that enables performance testing and analysis. In Section 4, we evaluate the performance of M-PKINIT and MP-PKINIT with both service time and queuing analytic models. After calibrating the model with our test configuration, we use it to support analysis and conclusions that extend beyond the skeleton implementation and development/test configuration. Finally, Section 5 provides a summary of our conclusions and outlines possible future work.

2. BACKGROUND ON MOBILE AUTHENTICATION APPLICATIONS AND THE USE OF PROXY SERVERS

Proxy servers have been widely studied and utilized [2, 3] in mobile computing architectures. The following paragraphs review two relevant examples that apply proxies to mobile authentication: Charon [3] and the Wireless Application Protocol (WAP) Wireless Transport Level Security (WTLS) [4].

2.1 Charon

The Charon protocol adapts standard Kerberos authentication to a mobile PDA platform. Kerberos is a well-known network authentication protocol. It uses secret key cryptography and relies on a trusted server, the Key Distribution Center (KDC), for the management of secret keys. There are many excellent overviews of Kerberos [5, 6] and a detailed protocol description will not be repeated here. Kerberos has had a recent increase in user base as the native network authentication protocol for Microsoft Windows 2000.

The transaction flow of Charon is presented in Figure 1. Charon uses Kerberos to establish a trust relationship between the user and the proxy. The mechanism for establishing this relationship is very similar to the method of establishing trust between the user and the target application server. As a result, there are several more interactions (i.e., network transmissions) required in Charon than in the standard Kerberos protocol, where there is no proxy involved. Charon uses the same encryption algorithms (i.e., DES) on the PDA as standard Kerberos. There is no network performance advantage using Charon versus 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 to assist in compute intensive operations. In [3], the Charon authors offer an example of potential areas for proxy assistance: retrieving email 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 Charon authors argue that the protocol can be proven correct since it does not change the basic semantics of Kerberos and Kerberos has been proven correct [7].

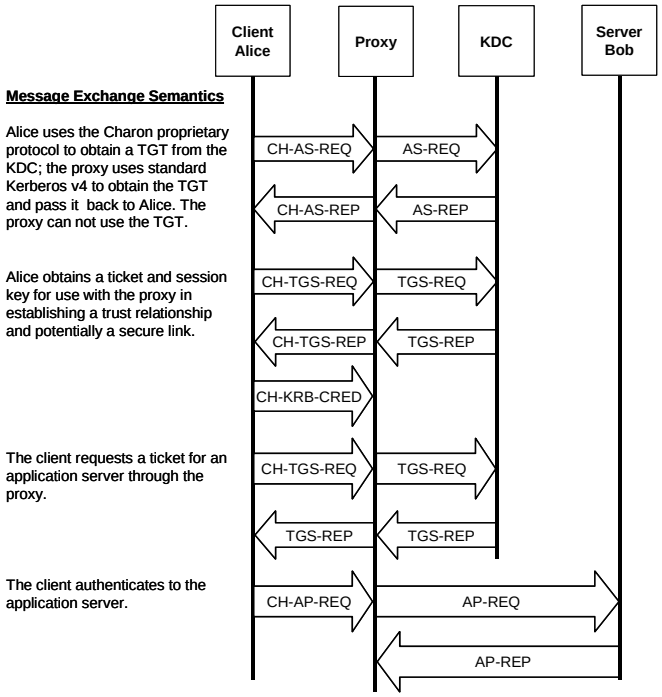


Figure 1. Charon Transaction Flow

2.2 WAP WTLS

WAP provides a lightweight set of protocols that allow resource-constrained computing platforms (e.g., cell phones) to operate on a data network such as the Internet. Since the WAP protocols are not directly interoperable with the traditional Internet, a gateway (i.e., proxy server) is required to perform protocol translation. The WAP version of TLS, the Internet protocol for performing authentication and establishing secure communications, is WTLS.

Figure 2 illustrates the WTLS transaction flow. 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 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 [8]. There are several objections to WTLS. For example, it allows the use of weak encryption algorithms and features that make chosen-plaintext attacks and brute force attacks easier to mount [9]. 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 cleartext messages. There are several proprietary and proposed standard solutions aimed at closing what has been called the "WAP Gap" [9] and implement end-to-end (i.e., mobile client-to-target server) security with WAP [11, 12].

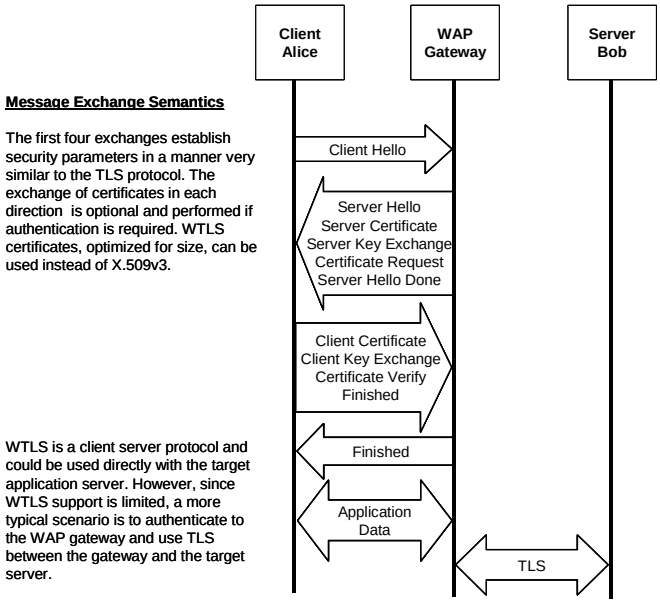


Figure 2. WTLS Transaction Flow

3. THE DESIGN OF PUBLIC KEY-KERBEROS AUTHENTICATION FOR MOBILE ENVIRONMENTS

Many current security architectures employ public key cryptography to implement confidentiality, integrity, and availability features. There are several proposals to add public key cryptography to the stages of Kerberos [1, 13-14]. The proposal to add public key cryptography to the initial user authentication in Kerberos is documented in an IETF draft and called PKINIT [1]. The PKINIT transaction is illustrated in Figure 3. Adapting PKINIT to a mobile computing platform would provide a mature authentication mechanism for mobile users. It would allow mobile users to operate in a public key infrastructure (PKI)-supported environment. However, there are at least two challenges in making this adaptation: (1) processing resource constraints on the mobile platform, and (2) communications resource constraints in the mobile network. These performance constraints could result in extended authentication response time for the user.

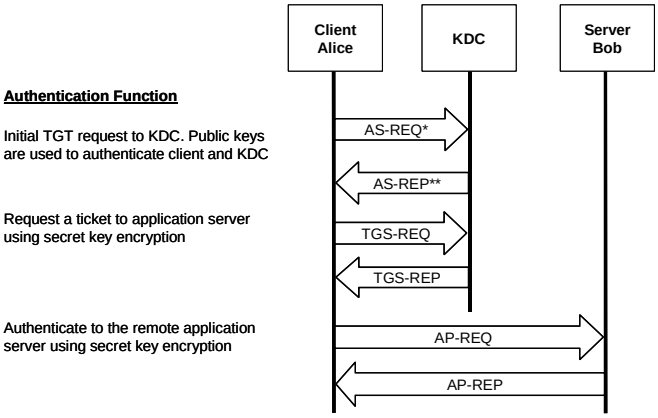
3.1 Design Guidelines

We began our formulation of PKINIT for a mobile platform by identifying a set of design guidelines. We have chosen guidelines that will potentially lead to lower response times and address some of the limitations identified in Charon and WTLS. The guidelines are:

- Reduce the number of public/private key operations performed on the mobile platform. In general, public key operations consume a significant amount of processing resources and will negatively impact performance and user response time. In some algorithms, and depending upon the encryption parameters, private key operations (e.g., signing) consume more compute resources than public key operations (e.g., signature verification). Minimizing the total number of public and private key operations will always improve performance. Designing the protocol such that the private key operation is executed on the mobile platform and the public key operation is executed on the KDC may improve mobile platform performance at the price of increased KDC workload.
- When a proxy is used, maintain the option to preserve the encrypted data stream through the proxy. A fundamental security criticism of WAP WTLS 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 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 message formats 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 correctness to be used in arguing that the new protocol is also correct.

3.2 PKINIT Adaptations in a Mobile Environment

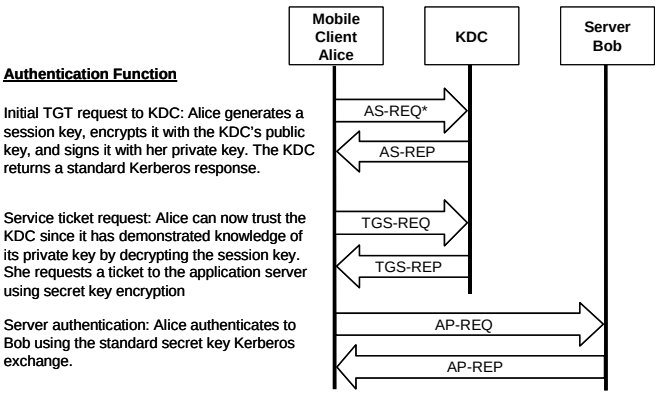
Figure 4 presents a mapping of the PKINIT protocol onto a mobile client, KDC, and target application server. We call this adaptation M-PKINIT. In M-PKINIT, there are only minor modifications made to the PKINIT protocol. One modification is to use an optional feature that appeared in a PKINIT draft that expired May 26, 1998, but was removed in more recent drafts. This feature accommodates the operation of PKINIT in situations where the client only possesses a signing key, but can also be used with RSA that 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, it is the KDC that 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 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.



* a standard Kerberos Version 5 message that contains a PA-PK-AS-REQ pre-authentication field including the userCert and SignedAuthPack

** a standard Kerberos Version 5 message that contains PA-PK-AS-REP pre-authentication field including the kdcCert, SignedReplyKeyPack, and encTmpKeyPack

Figure 3. PKINIT Transaction



* a standard Kerberos Version 5 message that contains a PA-PK-AS-SIGN pre-authentication field including the "userCert" and "encSignedRandomKeyPack"

Figure 4. M-PKINIT Transaction

Figure 5 presents a mapping of the PKINIT protocol onto a mobile client, proxy, KDC, and target application server. We call this adaptation MP-PKINIT. MP-PKINIT also uses the option that allows the client to generate the KDC session key to save a public key operation. 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 will encrypt the session key with the proxy's public key so that the discovery of the session key will require knowledge of the proxy private key. The proxy introduces an additional store and forward node into the protocol. This will certainly create more processing and communications overhead. To attempt to mitigate this overhead, an additional shortcut is taken: the client's certificate chain is cached at the proxy. This eliminates the need to transfer the client certificate(s) over the wireless network.

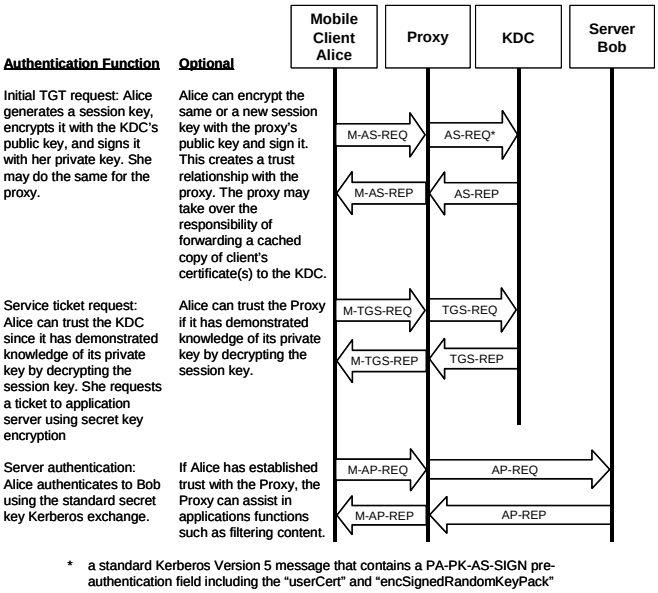


Figure 5. MP-PKINIT Transaction Flow

3.3 Skeleton Implementations of M-PKINIT and MP-PKINIT

In order to evaluate the performance characteristics of M-PKINIT and MP-PKINIT, we constructed skeleton implementations. The objective of the skeleton implementations is to consume processing and communications resources as would be expected in a full implementation while avoiding complexities such as error processing and optional features that would increase development time and not consume significant additional resources.

The development and test configuration is illustrated in Figure 6. The mobile device is a Vadem Clio C-1000 that 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 KDC and servers are all low-end Pentium processors running the Windows NT operating system. The configuration includes four conventional Windows 98 low-end Pentium client workstations to generate larger numbers of transactions and load the servers. We wrote the application in C++ and employed the RSAREF

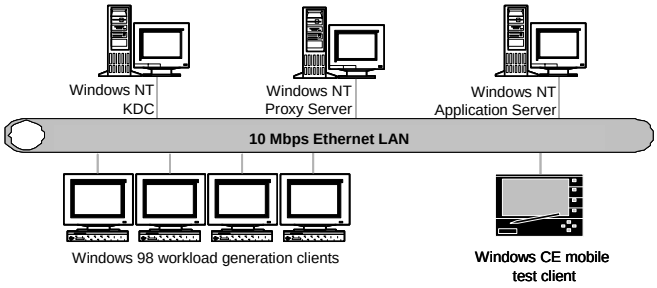


Figure 6. Development and Test Configuration

implementation standard software for public key encryption and the Karn portable C DES software for secret key encryption. All RSA keys were 1024 bits in length. We measured system performance by instrumenting the code and using an IP-level packet monitor.

The skeleton software architecture is illustrated in Figure 7. Two processes run on the proxy and Kerberos KDC. Processes P1 and K1 each open a TCP listening socket and wait for PKINIT transactions. Processes P2 and K2 wait for standard Kerberos requests arriving as UDP datagrams. The architecture follows the PKINIT IETF draft recommendations that the public key exchanges use TCP because of longer message sizes and the potential for message fragmentation. TCP connections are kept open as long as possible to reduce the effects of connection setup and slowstarts. For example, the proxy (i.e., process P1) holds its connection with the client open while it is communicating with the KDC on the client's behalf. The standard (i.e., secret key) Kerberos transactions use UDP as is common in current Kerberos implementations. All KDC, proxy, and application server processes are multi-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.

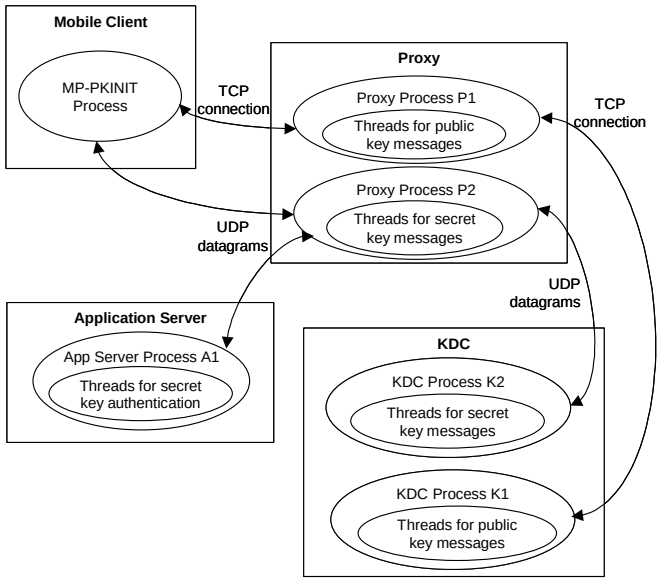


Figure 7. MP-PKINIT Skeleton Implementation

4. PERFORMANCE ANALYSIS

4.1 Service Time Measurement

The skeleton implementation and the test platform allowed us to measure no-load service time for a single authentication transaction under a variety of protocol permutations. The protocol permutations selectively engage subsets of the features proposed in Section 3. Table 1 summarizes the results. We recorded service times for six permutations: (1) *M-PKINIT standard* is the unmodified PKINIT protocol skeleton executed on the mobile client, a KDC, and an applications server; (2) *M-PKINIT client session key* adds the feature whereby the client generates the session key and trades a private key operation for a public key operation; (3) *MP-PKINIT standard* is the unmodified PKINIT protocol mapped onto the mobile client, a proxy, a KDC, and an application server. The proxy only performs store and forward operations; (4) *MP-PKINIT client session key* has the client generate the session key; (5) *MP-PKINIT trusted proxy* includes an authentication between the client and the proxy to establish a trust relationship; and (6) *MP-PKINIT proxy assist* has the proxy cache and add 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 broken up 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, and (4) *TGT & ST* includes all other client, Proxy, KDC, and application server processing related to the Kerberos ticket granting ticket (TGT) and service ticket (ST). Table 1 indicates that the Auth step always consumes the most time since it includes a significant amount of KDC processing, such as look-up of the client and application server in the Kerberos database, in addition to encryption functions.

Table 1. Service Time Measurements

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

Table 1 also illustrates the impact of the client generating 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 first 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 PDA service timesaving in the Post-auth step. Since only the KDC can decrypt the session key with its private key, the client can authenticate the KDC by simply confirming that the KDC properly encrypted the "EncKDCRepPart" [1] portion of the message. There are no further public key operations required in Post-Auth for the client to authenticate the KDC. For the test configuration, the increase in KDC service time results in the *M-PKINIT client session key* transaction producing 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 1), although the total response time is higher because of the additional delays 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 1). 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). This is because the message size savings that results from the proxy caching the client certificate is not significant at the local area network speeds of the test configuration.

The performance of the components of a typical mobile computing environment may vary significantly from our test configuration. In particular, the network connecting 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 server) will most likely be more powerful than the low-end Pentium workstations that we used for testing.

Figure 8 presents an analysis of sensitivity to server and wireless network capacity. The transaction component service times were varied in three ways: (1) we lowered wide area network throughput to 12,750 bytes/second to reflect typical Internet speeds [16]; (2) we used two wireless network rates—9600 bits/second to represent 2G and 384 Kbits/second to represent 3G network capacities [17]; and (3) we varied the server capacities by a multiplier ranging from 1 to 20 times the capacity of those in the test bed. At both wireless network speeds measured, the *M-PKINIT client session key* protocol variant performs sightly better than the *M-PKINIT standard* variant as the server speed-up hits the factor of ten range. 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 indicate that multipliers as high as 100 are appropriate if comparing the CPU capacity of a high end server to our test bed PCs. A more interesting result is that adding the proxy service, in this case, caching certificates for the client, bring the response times into a reasonable range at 2G speeds—reducing the overall response time from about 15 seconds to just above 8 seconds. This reduction is based on an average certificate size of 1.8KB, consistent with the range of sizes of commercial certificates [18, 19]. When the wireless network throughput is increased to 3G speeds, the proxy is a response time burden and increases the response time by over a second.

4.2 Closed Queuing Network Modeling

The service time analysis does not represent operation in a network in which resources are shared among many users. The KDC, proxy, and application servers are of particular concern. Authentication protocols that use public key encryption have been observed to consume a significant amount of server resources [20].

In [21], we developed a modeling strategy that used closed queuing networks with class switching [22] to represent public key variants of Kerberos under a variety of host and network assumptions. This technique allowed us to model transactions that consumed widely varying average service times at different visits to each queuing server. It permits modeling protocols that combine both public key and secret key encryption—potentially generating different average service times on each queuing server

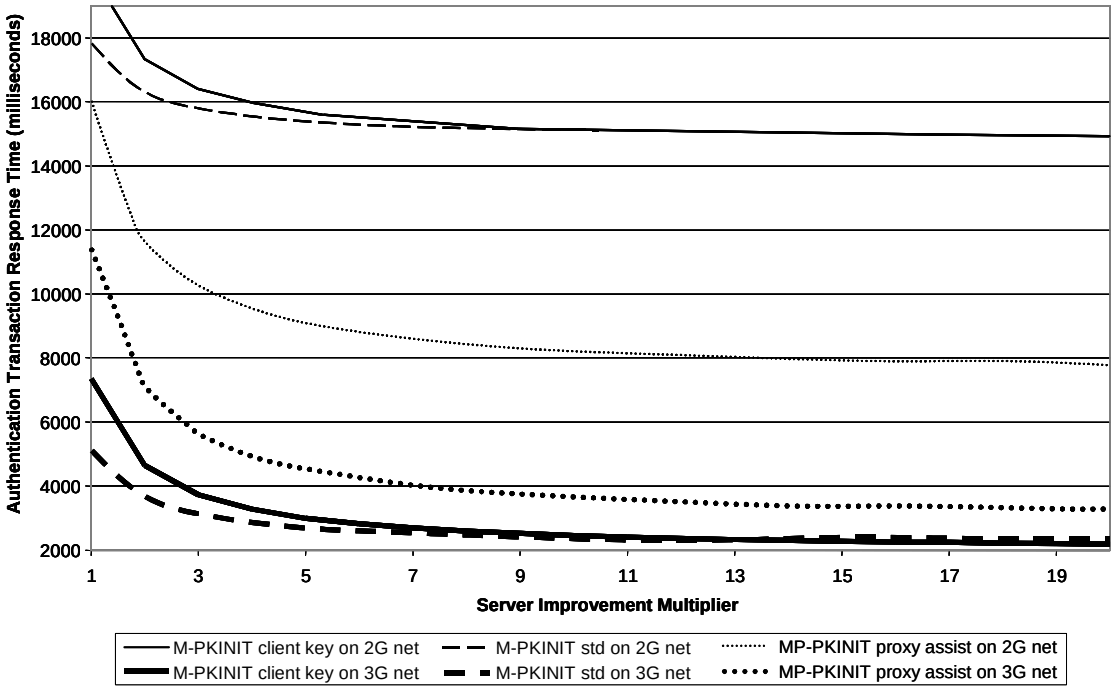


Figure 8. Sensitivity to Server and Network Capacity

visit. Using the class switching formulation, the queuing network maintains independent Markov chains for each closed group of classes and preserves the assumptions necessary to use fast solution methods [23, 24]. We used the Mean Value Analysis (MVA) technique with a Schweitzer approximation [25], resulting in very fast solution computation times.

Figure 9 presents the queuing network topology. 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. The topology anticipates that a wide area network (WAN), such as the Internet, connects the KDC and application server. The mathematical solution to the queuing network produces performance metrics for each queuing station and the system as a whole, such as the average number of customers, the average delay time, and the customer throughput. These metrics can be used to compare the performance of the alternative Kerberos adaptations.

Figure 10 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 to run in a 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. Figure 10 demonstrates good calibration between the model and observed test bed results supporting the model's predictive accuracy.

Figure 11 plots M-PKINIT and MP-PKINIT authentication transaction throughput versus response time, assuming a wireless network speed of 9600 bps and 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 is modeled as a fixed delay server—no increases in response time as a result of increased authentication traffic. We make 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 derive the throughput for the wireless network by degrading transmission speeds to account for frame errors using the measurements and analysis performed in [26]. 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.

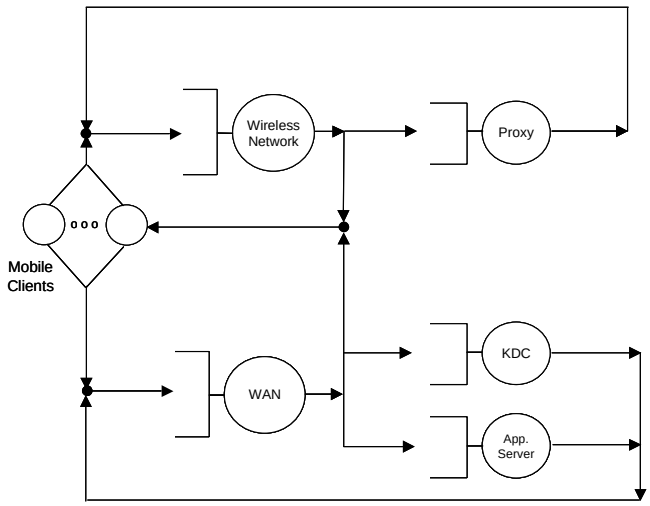


Figure 9. Queuing Network Topology

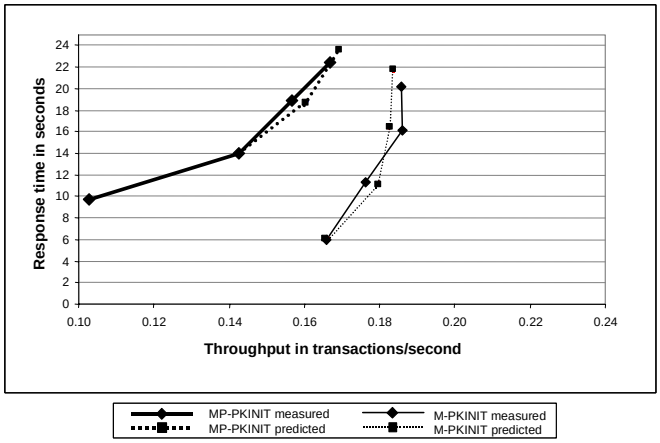


Figure 10. Model Calibration Results

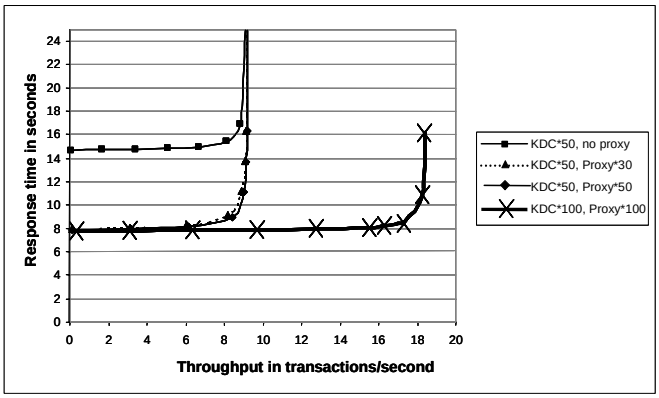


Figure 11. Server-Driven Performance Characteristics for M-PKINIT and MP-PKINIT

To underscore the role of the KDC, we decreased the speed up 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 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 non-proxy protocol curve knee occurs at the same place as the proxied version. This is because the KDC workload is the same for both proxied and non-proxied 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.

5. ANALYSIS AND CONCLUSIONS

5.1 Do M-PKINIT and MP-PKINIT Meet the Design Guidelines?

In Section 3 of this paper, we defined four design guidelines for the adaptation of Kerberos to a mobile computing environment. Does M-PKINIT and MP-PKINIT meet these guidelines? Specifically, did we:

Reduce the number of public key operations performed on the mobile platform? By using an optional feature of PKINIT intended for situations where 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 does not have to verify the KDC's signature since the KDC can only decrypt the client-generated session key with its 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 is used, 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. That 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—at the client's choice. MP-PKINIT provides the option for the client to preserve the encrypted data stream through the proxy.

Retain 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 there are several implementation details that must be addressed to make this work. An IETF draft [27] specifies a Kerberos protocol enhancement called IAKERB that 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 if the KDC is not the bottleneck and it may not be worth changing current Kerberos implementations to support this feature. Beyond these two areas, M-PKINIT and MP-PKINIT can both employ standard Kerberos message formats at the interface to the KDC and application server.

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 a re-formulation of the Kerberos formal proof before one could assert that M-PKINIT and MP-PKINIT are provably correct adaptations of Kerberos.

Our measurements and models have demonstrated that public key Kerberos can be adapted to a mobile environment and with suitable KDC performance, can provide reasonable user response times (i.e., approximately 8 seconds). We achieved acceptable performance with a well-proven public key encryption algorithm: RSA with 1024-bit keys. At G2 wireless network speeds (i.e., 9600 bps), assistance is required from a proxy server in order to produce adequate response times—the proxy caches certificates for the client to reduce the wireless network message traffic. The current IETF draft for PKINIT allows the KDC to store client private keys. While this would also eliminate the need to transmit certificates, it would reduce the general applicability of the protocol to situations where the client was pre-registered with the KDC.

5.2 Future Work

Through our skeleton implementation and analytical models, we have demonstrated that assistance from a proxy server makes our adaptation of public key Kerberos a viable authentication protocol

at 2G wireless network speeds. There are several areas where further work may be conducted.

Our skeleton software uses the RSAREF public key encryption utilities. While RSA algorithms are common in public key applications, there are other ciphers that are also of interest for authentication. The PKINIT draft cites the Diffie-Hellman key establishment algorithm as a mandatory implementation requirement. In addition, there has been significant interest and use of elliptic curve-based encryption algorithms in mobile computing environments. Elliptic curve algorithms are believed to provide comparable assurance levels to RSA with a smaller key and, as a result, a less computationally intensive calculation. The extension of the analysis presented by substituting Diffie-Hellman and elliptic curve for RSA remains as future work.

The test and model configurations both assumed a mobile client and a stationary KDC and application server. A variation of this configuration might include KDCs and applications servers that are also mobile. We envision that such a configuration might be found, for example, in a military battlefield environment. This variation potentially reduces the speed of communications between the proxy and the KDC and the benefits that accrue from proxy-cached certificates. Further analysis of a mobile KDC and proxy remains as future work.

We have shown that the class switching queuing formulation is an effective way to quantitatively analyze the performance of protocols that combine secret and public key cryptography. The application of this analysis technique to a broader range of protocols will also be the topic of future research.

Finally, our specification of M-PKINIT and MP-PKINIT, as well as the original PKINIT protocol, does modify the semantics of Kerberos. Extension of the work in [7] to demonstrate the correctness of M-PKINIT and MP-PKINIT protocols remains as future work.

6. REFERENCES

[1] Tung, B., *et al.*, *Public Key Cryptography for Initial Authentication in Kerberos*, 2001: http://www.ietf.org /internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt.

[2] Zenel, B., *A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment.* Wireless Networks, 1999. **5**: p. 391-409.

[3] Fox, A. and S.D. Gribble. *Security on the Move: Indirect Authentication Using Kerberos.* in *MOBICOM 96*. 1996. Rye, New York.

[4] Wireless Application Forum, Ltd. 2000, *Wireless Application Protocol Wireless Transport Layer Security Specification*, WAP-199-WTLS, February 18, 2000.

[5] MIT, *Kerberos: The Network Authentication Protocol*, 1998, http://web.mit.edu/kerberos/www/.

[6] Kaufman, C., R. Perlman, and M. Speciner, *Network Security, Private Communication in a Public World*. 1995, Englewood Cliffs, New Jersey: PTR Prentice Hall.

[7] Burrows, M., M. Abadi, and R. Needhan, *A Logic of Authentication.* ACM Transactions on Computer Systems, 1990. **8**(1): p. 18-36.

[8] Khare, R., *W* Effect Considered Harmful*, 1999, 4K Associates.

[9] Jormalainen, S. and J. Laine, *Security in the WTLS*, 1999, Helsinki University of Technology: Helsinki.

[10] DeJesus, E.X., *Locking Down the...*, in *Information Security Magazine*. 2000.

[11] Cylink, *"Closing the 'Gap in WAP'"*, 2000.

[12] WAP, *Wireless Application Protocl TLS Profile and Tunneling Specification*, 2000.

[13] Medvinsky, A., *et al.*, *Public Key Utilizing Tickets for Application Servers (PKTAPP)*, 1997: http://www.ietf.org /internet-drafts/draft-ietf-cat-kerberos-pk-tapp-03.txt.

[14] Hur, M., *et al.*, *Public Key Cryptography for Cross-Realm Authentication in Kerberos*, 2000: http://www.ietf.org /internet-drafts/draft-ietf-cat-kerberos-pk-cross-06.txt.

[15] Tung, B., *et al.*, *Public Key Cryptography for Cross-Realm Authentication in Kerberos*, 1998: http://www.internic.net /internet-drafts/draft-ietf-cat-derberos-pk-cross-03.txt.

[16] Menascé, D.A. and V.A.F. Almeida, *Capacity Planning for Web Performance*. 1998: Prentice-Hall Inc.

[17] Personal Communications Industry Association, *Market Demand Forecast for Terrestrial Third Generation (IMT-2000) Service for the Peronal Communications Industry Association*, 1998.

[18] Taschler, S., *Datakey CIP 3.0 Whitepaper*, 1997.

[19] *Consideration of Smart Cards as the DoD PKI Authentication Device Carrier*, 2000, Office of the Secretary of Defense.

[20] Apostolopoulos, G., V. Peris, and D. Saha. *Transport Layer Security: How much does it really cost?* in *IEEE INFOCOM*. 1999.

[21] Harbitter, A. and D.A. Menascé. *Performance of Public Key-Enabled Kerberos Authentication in Large Networks*. in *IEEE Conference on Security and Privacy*. 2001. Oakland, California.

[22] Bruell, S.C. and G. Balbo, *Computational Algorithms for Closed Queueing Networks*. The Computer Science Library, ed. P.J. Denning. 1980, New York: Elsevier North Holland, Inc.

[23] Menascé, D.A., V.A.F. Almeida, and L. Dowdy, W., *Capacity Planning and Performance Modeling*. 1994, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

[24] Gross, D. and C.M. Harris, *Fundamentals of Queuing Theory*. Third ed. 1998, New York: John Wiley & Sons, Inc. 439.

[25] Schweitzer, P.J., *A Survey of Mean Value Analysis, its Generalizations, and Applications, for Networks of Queues*, 1991, William I. Simon Graduate School of Business Administration, University of Rochester: Rochester, NY.

[26] Xylomenos, G. and G.C. Polyzos, *Internet Protocol Performance over Networks with Wireless Links.* Mobicom 99, 1999.

[27] Swift, M., *et al., Initial and Pass Through Authentication Using Kerberos V5 and the GSS-API (IAKERB)*, 2001, IETF.