Enhancing Kerberos security using Public key and Context-aware Access Control

Gauri Gunjal[#1], S.K.Pathan[*2]

[#,*] *Department of Computer Engineering, Smt Kashibai Navale College of Engineering*

Affiliated to University of Pune, Maharashtra, India

Abstract: **Authentication is a first phase in standard Kerberos which is password dependent. It makes the protocol vulnerable from attackers for password hacking using offline or online attacks. In this paper we have proposed password-less authentication which is based on public-key that gives increase security but at the same time it's lighter on computations & network traffic compare to well-known PKI. Authorization is second phase of kerberos. Here technically kerberos provides service ticket to all valid users present in system & expects Application Server to take decisions on granting or rejecting request during third phase. In this paper we have customized second phase of kerberos so that authorization decision for critical services can be taken by using user's context data. This enhancement would stop sending service tickets to unauthorized users making hackers job really tough to reach onto third phase. The context aware control mechanism we proposed here addresses core security needs of any organization who wants to tightly control access to critical services.**

Keywords: **Authentication, Authorization, Context-Aware authorization, Kerberos authentication protocol, Public Key Systems.**

I. Introduction

Kerberos Authentication Protocol works purely on tickets. It prevents clear-text passwords from being sent across the network. Kerberos supports standard ACL (Access Control List) where we can carry extra attributes for authorization and we may independently implement a totally new fine-grained access control mechanism. Kerberos's strong security against eavesdropping, and support for mutual authentication motivates us to come up with a newer approach to make it more secure and suitable for organizations who wants to control user access to critical services.

Despite Kerberos many strengths, it has some weaknesses and limitations as below:

- Kerberos works on symmetric key. Kerberos server store user's id and passwords. There is fair possibility that by gaining access to kerberos database attacker can get unauthorized access.
- Kerberos server does not verify user's identity and simply returns tickets to the requester that includes strongly encrypted ticket. This means intruder or hacker may gather many tickets of many users to perform offline dictionary attack.
- Kerberos supports only course-grain authorization. It does not have strong validation or rejection mechanism while providing Service Tickets to the user.

This paper mainly focuses on integrating public key and location based security features into Kerberos. It attempts to improve security during user authentication phase to eliminate offline dictionary attacks. For service requests we try to build validation & rejection mechanism by allowing user's context information to be sent to Kerberos server and providing dynamic authorization capability based on the user's runtime parameter.

The remainder of this paper is organized as follows: A brief review of Kerberos related work is given in section II. Section III describes proposed framework, core design, data flow and mathematical steps. Section IV and V details proposed Authentication phase and Authorization phase respectively. Section VI explains about experimental setup and implementation. Results are presented in section VII. Paper is concluded in section VIII.

II. Related Work

Kerberos was developed by MIT for authentication as a part of Project Athena [1, 2]. All the major operating systems and known vendors refer Kerberos as Secure Authentication Protocol. These includes but not limit to Windows, FreeBSD, Apple's Mac OS X, Red Hat Enterprise Linux, Oracle's Solaris, IBM's AIX. Authentication using Kerberos protocol are always reusable and durable for the lifetime of the ticket. It uses duel key encryption methodology which makes communication between client and server completely secure. It provides single sign-on and mutual authentication features.

Password-guessing, brute force attacks are not solved with Kerberos. The KDC must be kept secure, both from a physical and a network standpoint. Kerberos does not itself provide authorization, but V5 Kerberos passes authorization information generated by other services. As original kerberos is based on symmetric key cryptography it does not scale well in large distributed environments. It also does not provide non-repudiation services like digital signatures.

In 1995[3], Asymmetric cryptosystem in a kerberos protocol is proposed. That is more secure version of kerberos with minimal changes to the protocol. RSA private key is split into two portions. One portion becomes a user's Yaksha password, and the other the Yaksha server's password for that user. But this system is dependent on another sub-system that temporarily generates private-public pair of passwords, which adds maintenance and separate monitoring work.

In 2010[4], Fast Algorithm is proposed to improve efficiency of Kerberos protocol. It also succeeds in preventing replay attacks as it uses random number but not the timestamp. However all communications between KDC, Client and Application Server are done through asymmetric ways.

PKINIT extensions [5] to RFC4120 support the use of public-key cryptography in the initial request for a ticket. In addition, the use of symmetric cryptography after the initial exchange is preferred for performance. Thus public-key based protocols like PKINIT, PKCROSS adds public-key support at different stages of the kerberos framework. All proposals of this type improve scalability and security by simplifying key management. But the computational requirement of PKC is higher which impacts performance [6]. Identity-based signcryption is proposed in 2013[7]. It's designed to eliminate the need of PKI and uses identity based signcryption to encrypt and sign messages. It also restricts attacker from examining message details.

Location-based authorization concept was officially presented and discussed for the first time in 1998 by Denning [8]. Denning mentioned how GPS signals can be used to locate the device and use this data to synchronize and authenticate client and server against each other. Location-based kerberos authentication protocol is proposed in 2010 [9]. Here user's physical position is added to the kerberos protocol's messages as a new factor. But this solution mainly focuses on the reducing the possibility of replay attack during authentication, it does not integrate location with authorization decisions.

In 1996 [10], Sandu introduces Role Based Access Control (RBAC). Here permissions are linked with roles than users. Microsoft has developed a newer approach through Dynamic Access Control capabilities [13]. IT's introduced in Windows Server 2012 and Windows 8. For example even if user has access to XYZ resources but user's device (like laptop, palmtop or smartphone) is in restricted list user won't get access to resource. Despite of dynamic provision system deals with static objects means it’s not truly dynamic and is not influenced by user or system's context information like location, time and situation (critical/non-critical).

In 2009, [11] a survey focused on location-aware authentication, including context-based policies has outlined many challenges in this area. As per this work is needed to develop systems which will be able to take into account context information about users. Efforts spent on Authorization so far focus on relatively static scenarios where access depends on identity of the subject.

In this paper we are trying to use public key cryptography for initial authentication phase and user's physical environmental context (current location) factor for dynamic authorization which provides end-to-end security.

III. Proposed PKCA-Kerberos Model

A. *Proposed PKCA-Kerberos Framework*

Below is proposed framework as shown in Fig. 1. At broad level it is divided into two parts: 1) User space and 2) Server space.

1) Users space: It consists of User’s machine which can be a desktop or laptop. User also carries separate utility device like smart card to keep his Private Key/Public Key separately. User is provided with a preconfigured Custom GPS Device called C-GPS. C-GPS carries unique device id which is pre-registered in organizational database and assigned to the user explicitly. This device acts as user’s context which shares data during user authorization phase.

2) Server space: It consists of Authentication Server, Ticket Granting Server as a part of KDC. KDC uses a customized LDAP database. Database consists of user data and service data. We have introduced new fields into the database like provision for storing user’s public key, user’s device key. Services are also identified as critical or non-critical. A Context Manager (CTXM) component has been introduced as shown in Fig. 1. CTXM can interpret encrypted context information (e.g. user’s locations which are produced by C-GPS device). Upon request from TGS it decrypts and then validates inputs against Context Database and verifies it with restriction policy. If the context is found outside policy or invalid then it returns error message to TGS.

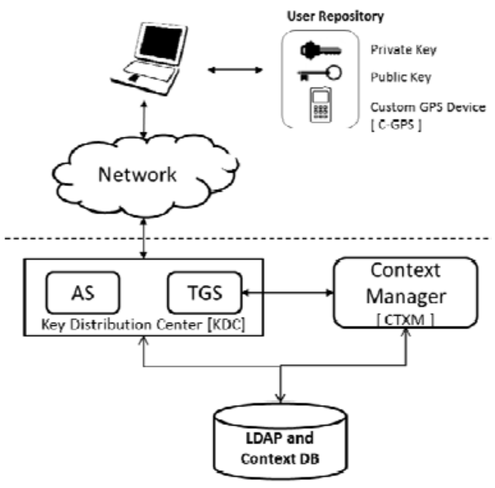

Fig. 1 High Level Architecture of PKCA-kerberos System

IV. User Authentication Phase

In traditional kerberos V5 during logon server really do not care about user's identity and returns a strongly encrypted tickets but as the encryption is finally linked with user’s password there is fair possibility that attacker can perform offline attacks. To eliminate risk associated with password hacking we have designed a new authentication phase which is lighter on computation than PKI but is also satisfies safety & security needs of Authentication Phase.

A. *PK authentication phase*

In this phase the client C creates authentication request called as AS_REQ. C uses pre-authentication (pre-auth) field of AS_REQ to pass its public key (PU_c). It also encrypts timestamp (TS) using its own private key (PR_c). This AS_REQ is then sent to Authentication Server (AS). Upon receiving the request AS checks if pre-auth field has user’s public key present in it. AS then compares this PU_c against the one present in LDAP database for same client. If key matches then AS generates authentication reply (AS_REP). This reply consists of 3 important elements which are ticket granting ticket (TGT), Random Key (RK) and Session Key ($K_{c,\,tgs}$). Here R_k is encrypted by AS using client’s public key PU_c.

- PU_c = Public Key of Client
- PR_c = Private Key of Client
- RK_c = Random Key
- Loc_c = Location of User

- ID_c = ID of Client
- ID_{tgs} = ID of Ticket Granting Server
- ID_v = ID of Application Server
- $K_{c,tgs}$ = Session Key for Client & TGS
- $K_{c,v}$ = Session Key for Client & Application Server
- TS = Time Stamp
- $AUTH_c$ = Authenticator
- TGT = Ticket Granting Ticket
- SGT = Service Granting Ticket

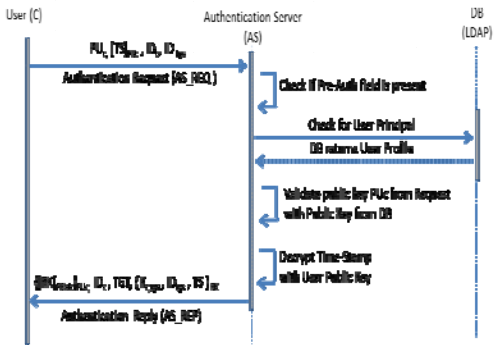
User (C)
Authentication Server (AS)
DB (LDAP)
PU_c, [TS]_PRc, ID_c, ID_tgs
Authentication Request (AS_REQ)
Check if Pre-Auth field is present
Check for User Principal
DB returns User Profile
Validate public key PUc from Request with Public Key from DB
Decrypt Time-Stamp with User Public Key
Authentication Reply (AS_REP)

Fig. 2 Information Exchange during Authentication phase

Fig. 2 shows sequence diagram for initial authentication phase. We call this phase as password-less because we have used user's public key for encrypting the random key in AS response and there is no use of user password in AS_REQ and AS_REP.

Client sends AS_REQ to Authentication Server

$C \rightarrow AS$: PU_c, $[TS]_{PRc}$, ID_c, ID_{tgs}

Authentication Server sends AS_REP to Client

$AS \rightarrow C$: $\{[RK]_{PRkdc}\}_{PUc}$, ID_c , TGT, $\{ K_{c,tgs}$, ID_{tgs} , TS $\}_{RK}$

TGT=$\{ K_{c,tgs}$,ID_c, ID_{tgs}, TS$\}_{Ktgs}$

Proposed authentication phase is also thin when we compare it with standard PKINIT.

B. Challenges with Standard PKI & proposed modifications:

Standard PKINIT relies on client certificate, server certificate, digital signature, message digest. Having digital certificates in request requires additional parameters like certificate authority, certificate lifetime, certificate directory, revocation list. Both the client and the KDC have a public-private key pair in order to prove their identities to each other over the open network. Intention of PKI infrastructure is to address issues of managing secret keys of large number of clients and secure authentication procedure by building new trust model where KDC is not the first entity to identify user. However for scenarios where revocation or suspend of certificates happen, legislation of digital signatures and related PKIs becomes difficult and challenging tasks.

We have proposed thin PKI model to address some of issues related to Standard PKI. We have reduced number of pre-authentication data elements used during AS_REQ. Thus it keeps message size between request and response as minimum as possible. PKINIT expects dataset consisting Auth-Pack, trusted certificates. We have not used these parameters as our framework assumes client & server carrying copy of each other's certificate which can be referred on demand from their database. We have eliminated computational overheads like certificate verification by client and server both. We propose maintenance of keys, certificate & updates in LDAP as offline activity outside authentication phase. Table I shows the comparison between PKINIT and our proposed PKCA-Kerberos.

TABLE I: Authentication Phase Comparison between PKINIT and PKCA-Kerberos

Parameter	PKINIT	PKCA-Kerberos
Keys Generation	Keys needs to be generated by a certificate authority (CA) and may involve cost behind each key-pair & certificate generation.	Keys can be generated by either a certificate authority (CA) or by using standard utilities which comes with Java/JDK to save the cost.
Privacy and Secrecy of user keys	Easy to trust, secured and proven model in the market. E.g. VeriSign, Entrust	Either organization can invest on maintaining proprietary setup for key generation thereby avoiding handling of key data outside organization's private network.
Dispatch & deployment of Private/Public Keys to the User Device	It introduces handling & physical transfer of keys data once generated by vendor to organization and then to the device. More handling adds possibility of cloning of private key outside organizations reach.	Handling of keys limits within organization boundary. So secure deployment of keys to context device is possible to achieve with lesser cost on processes, software's and peoples.
Publishing and Revocation of Certificates	Publish period for updates or revocation is usually in weekly cycle. This means Certificate Services and the organization system may go out of sync, which may give opportunity for disabled user to do intrusion within stipulated time (i.e. time between enablement and disablement)	Having LDAP database and keeping a separate attributes to track user's active or inactive status eliminates dependency on Certificate Service for validity of users. This means user's access and authorization can be disabled or enabled instantly.
Certificate Transfer during AS Exchange	Copy of User Certificate and KDC Certificate are transferred to and fro during authentication. If a organization has 10,000+ users and assuming user logs in two times in a day: User certificates transmits for AS exchange = 20000. KDC certificate transmits for AS exchange = 20000. Means there is transfer of 40000 certificates over wire in a day. This definitely act as overhead to the system especially when organization has large employee population.	Certificates do not get transmitted during kerberos request or response. System relies on preinstalled certificates in database & client's device. Any updates to certificates, keys can be done offline as on needed. This saves the network bandwidth used for Authentication across users.
Certificates Use	Certificate management in traditional public key infrastructure (PKI) is inefficient	No certificates used in AS_REQ/AS_REP hence •Low communication bandwidth •No need to verify certificates online (certificate chains)

We also verified our new framework design to see if it will be sustain popular attacks. While designing the new framework we tried to improve protocol such that attacks impacting Kerberos V5 protocol can be either minimized or eliminated.

Kerberos is vulnerable to password guessing attacks. In case if user chooses a password which is poor then for an attacker it is easy to attack. He can easily mount an offline dictionary attack where he can pick up entries from dictionary and repeatedly attempting to decrypt the messages obtained during hacking. As message is originally encrypted using a key derived from password it is probably easy for hacker to finally get the key through trial and error. In case of brute force attack attacker uses exhaustive procedure and tries all possibilities one-by-one to crack the user's password.

Password guessing and brute force attack are totally eliminated as in our implemented approach there is no need of user's password for authentication, which has been replaced by public key cryptography. Table II shows the comparison of possible attacks on Kerberos V5 and Proposed PKCA framework.

TABLE II: Comparison of Possible Attacks

Possible Attacks	PKCA-Kerberos	Kerberos V5
Password guessing	✓	✗
Dictionary Attacks	✓	✗
Brute Force Attack	✓	✗

#Notations: ✓ non-vulnerable: ✗ vulnerable

V. User Authorization Phase

We have modified user authorization phase in to two parts. First part is called as context generation phase. This phase is to generate user's context using Context Device Custom GPS. This context data is then embedded in to TGS_REQ by user's kerberos client and is sent to server for authorization verification.

A. *User Context Generation Phase*

Fig. 3 shows sequence diagram for user context generation phase between user machine and CGPS device. Upon user's request kerberos client process sends notification to Context Device CGPS requesting new user context using USER_CONTEXT_REQ message. CGPS then calls GPS service and collects GPS location. To make this phase secure we assume that CGPS device has capability to encrypt the location and current timestamp using device key which is unique. This data is collected in USER_CONTEXT_REP. This data is then shared with user's kerberos client.

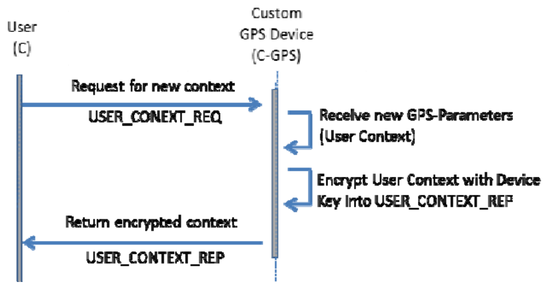

Fig. 3 Information Exchange during User Context Generation Phase

B. *New context based authorization phase*

Upon receiving encrypted USER_CONTEXT_REP from CGPS device, kerberos client then generates service request called as TGS_REQ as shown in Fig. 4. This request mainly consist of application server's ID (IDv), TGT and Authenticator. As authorization request needs dynamic data from user, we need to share USER_CONTEXT_REP in TGS_REQ. As per Kerberos RFC 4120 has provision in authenticator to pass additional authorization data. USER_CONTEXT_REP added to authenticator field of TGS_REQ which gets encrypted using shared session key Kc,tgs.

At the server end, TGS revives user request. It performs basic validations and it then checks for presence of context data in Authenticator field. It decrypts authenticator field using Kc,tgs and retrieves USER_CONTEXT_REP. Upon successful decryption TGS server then sends user principal, target server name and USER_CONTEXT_REP to Context Manager CTXM. Main role of CTXM manager is to decrypt USER_CONTEXT_REP using secret key from Context DB. If decryption is successful, it then checks the context is within allowed limits as per Organization Policy Rules. If this step is successful then CTXM returns success message to TGS. TGS in turn generates response TGS_REP and returns it back to Client.

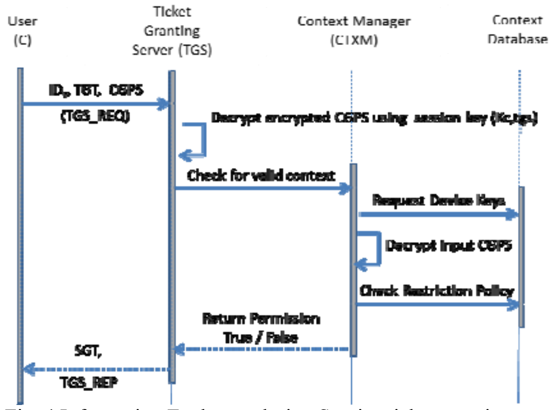

Fig. 4 Information Exchange during Service ticket granting

Client sends TGS_REQ to Ticket Granting Server
$C \rightarrow TGS: ID_v, TGT, AUTH_C$

Ticket Granting Server sends TGS_REP to Client
$TGS \rightarrow C: ID_c, SGT, \{Kc,v, ID_v, TS\}_{Kc,tgs}$

$SGT = \{K_{c,v}, ID_c, ID_v, TS\}_{Kv}$
$AUTH_c = \{IDc, TS, CGPS\}_{Kc,tgs}$

C. *Overcoming limitations in Traditional Kerberos Authorization Phase*

Traditional kerberos authorization phase takes minimal part in authorization. If user sends a valid SGT_REQ then KDC generates service ticket SGT and returns it back to the user. Here it is expected that the target Application Server is capable of taking authorization related validations.

In case an organization wants to tightly control access for critical services using traditional kerberos system, then the key authorization checks needs to be applied on Application Servers. But Application Server comes into action only in last phase of kerberos. This means if the organization has 'n' critical services then all of this services will be deploying authorization checks at their end. Having multiple servers involved makes it difficult in maintaining authorization policy for critical services of the organization.

In this paper we have overcome above problem by enhancing phase-II of kerberos. Here Ticket Granting Server and Context-Manager works together to validate user's context, performs additional checks like verifying if user's context is within restricted zone. This makes 2nd phase stronger & addresses organization's key need to centralize important authorization decisions. If user is unable to prove its context TGS will not provide service ticket SGT. Not having SGT in hand makes it impossible for the user to access the target service because his request will be immediately rejected in 3rd phase of kerberos.

In traditional kerberos as TGS returns SGT to all legitimate users, it increases possibility of 'Replay' attack in 3rd phase. However in our proposed framework, SGT is not return to the user unless he passes through critical authorization validation. Not receiving a SGT makes it difficult for hacker to do replay attack in 3rd phase of kerberos.

D. Comparison between proposed PKCA-Kerberos and PKINIT based authorization systems.

In this paper we have come up with a framework to address some of the limitations in Kerberos protocol. Table III presents a detail comparison between standard PKINIT and PKCA framework against various parameters.

TABLE III: Comparison of Authorization Phase between PKINIT and PKCA-Kerberos

Parameters	**PKINIT**	**PKCA-Kerberos**
Authorization approach	Distributed across Application Servers	Centralized in to KDC Domain
Supported functions with TGS Server	Basic only. TGS grants SGT to all legitimate users.	Enhanced. TGS performs detail authorization checks.
Dynamic authorization capabilities	No	Yes – As system captures user's run time context securely.
Dependency on Application Server during Authorization	Necessary. It's expected that Application Server handles required authorization checks in phase III.	Independent of Application Server. TGS & Context Manager together can restrict users from granting service tickets for critical services in phase II.
Implementing new authorization policies for critical services	Costlier and time consuming as it's decentralized. It impacts all critical services. All serveries need to be modified and tested to fulfil new authorization requirements.	It's less costly comparatively. This is because many authorization decisions can be centrally managed at TGS and Context Manager irrespective of number of critical services present in the network.

Replay attacks are usually very dangerous as attacker gains direct access to resources without tampering the data. This means to safeguard critical services it important to protect the information from middle man and hackers. Kerberos V5 cannot prevent replay attack in 3rd phase this is because in 3rd phase only Client and target service or target application server comes in to picture. Client holds SGT and here hacker can replay request and possibly get access to the critical service. For organizations this becomes critical problem.

In our implemented approach we have introduced context based authorization which is a check on user's context (like location) accordingly decision is taken if SGT should be return to user or not. As user will not able to acquire SGT in second phase it stops unauthorized user to avail SGT. Not having SGT eliminates opportunity of unauthorized user's accessing the target application server.

Attack scenario at server side: Single point of failure. It requires continuous availability of central server. In our approach we have used Kerberos DB for storing user's public key which is required for verification of user's identity. When the Kerberos server is down no one can log in. this can be solved by using multiple servers.

VI. EXPERIMENTAL SETUP AND IMPLEMENTATION

In this section, we have presented the prototype implementation of proposed PKCA-Kerberos framework. Server side modifications are done on ApacheDS™ 2.0.

ApacheDS is an extensible and embeddable directory server entirely written in Java, which has been certified LDAPv3 compatible by the Open Group. Besides LDAP it supports Kerberos 5 and the Change Password Protocol.

A. PKCA Kerberos Experimental Setup

The proposed framework is based on kerberos protocol. As shown in Fig. 5 prototype implementation is created in local environment which consist of Apache Directory Server, Local Client, GPS Device Simulator, LDAP Database, Test Application Server. We have used Apache Directory Server which has inbuilt support for KDC server as per Kerberos V5 specifications. Apache Directory Server uses LDAP protocol where we store user information, device information, and application server information. Context Manager Component is also written in java and is integrated with KDC Server for testing purpose. We have written a standalone kerberos client which is composed of authentication and authorization request components and GPS device simulator written in java.

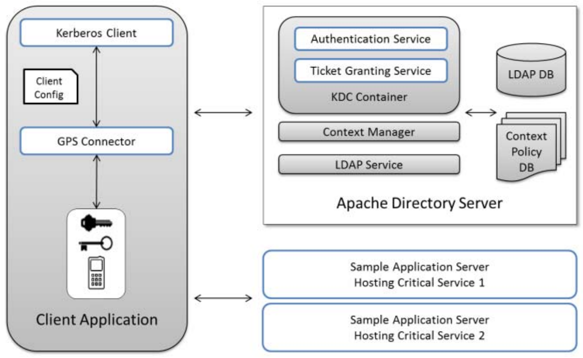

Fig. 5 Experimental setup of PKCA-Kerberos System

1. Setup Users: This is done using LDIF file which includes all user attributes. It gets imported in LDAP database using Apache Directory Utility.
2. Public and Private Key Generation: Command line JDK based keytool is used for this. keystore.jks file is generated which act as repository for storing security certificates.

3. Context Device setup: For project purpose we have assumed that there is a custom context device that has context private key securely burned into device hardware. Each device key is registered in LDAP against unique user.
4. Access Restriction Policy Setup: We have created a policy file which stores allowed locations, regions, co-ordinates as a part of restricted access area setup.

B. *Implementation*

In logon phase, A legitimate user sends authentication request by entering user id through kerberos client terminal. Kerberos Client software performs client side computation. It sends the request to the KDC server. Authentication Server Component of the Apache DS validates the input request, verifies public key in request with the one in LDAP database. It also decrypts encrypted timestamp to see if the request has come from legitimate user or not. After all validation tests are successful it returns KDC reply to the client. Client stores this response part for further service requests till the ticket is valid.

Let’s assume user wants to access FTP service which is one of the critical services. In this case user activates GPS device (simulator) which fetches current physical location of the user, it then encrypts it with device key and returns it to the Kerberos Client Application. Kerberos Client application then sends this authorization data in service request headers to the Authorization Server Component of KDC. Authorization Server then decrypts request, sends the encrypted location to Context Manager for further authorization verification. If user’s location is within allowed regions Context Manager returns success flag to Authorization Server Component. Server then returns SGT ticket to the user successfully.

VII. Result Summary

As mentioned in proposed system, during second phase of kerberos system checks for user' context attributes received in request. TGS server then uses server side components to verify user's eligibility for given service and decide if user should be given with service ticket or not. Thus we build a strong protection model into Kerberos which will ultimately eliminate third phase when user do not have required privileges to access the service. This means there is reduction in total number of kerberos transactions, bandwidth and time. This benefit is more visible when we apply this framework for organizations having employees in thousands (e.g. 100k). Almost every employee logs in to system & accesses services each day. It saves CPU consumption on Application Server, it also makes authorization system easily scalable & there is less work on the Application Server. But in standard kerberos application Server had to decrypt the service ticket and is fully responsible to performs authorization checks for user explicitly.

With new PKCA-Kerberos System tickets are generated only when user is completely authorized to access the service, which is not the case with original Kerberos V5. Not generating Service Tickets for such cases eliminates unnecessary ticket transmission between Client & Server. It would make hackers job difficult (e.g. man in middle, offline attackers, replay attacks) as they will not have any handlers to tickets

We assume that for a smaller organization there exist two services S1 –ftp, S2 – telnet which are critical in nature. To avoid business impact this organization wants to provide access for remote users. But on other hand it also wants to limit access from specific regions due to their working related regulations & company policy.

TABLE IV: Users and Service Access mapping for statistical analysis

Group	Users	Access Grants
G1	30	S1 only
G2	30	S2 only

In Kerberos V5 if all users from G2 groups try to get service ticket for S1 service, KDC will return SGT for all because it will expect the Application Server (S1 or S2) to handle the authorization decisions. This means SGT is generated for each request whether it is for S1 or S2 server. It doesn’t care if the requester belongs to G1 group or G2 Group. In all there are 120 tickets created using traditional Kerberos V5 server. Also, as Kerberos V5 is not handling dynamic data like context data, its returning tickets to all users provided all are authenticated successfully.

No of Tickets with Kerberos based system:

Group	User Requests	SGT Tickets
G1	[G1, S1] + [G1, S2]	30 + 30 = 60
G2	[G2, S1] + [G2, S2]	30 + 30 = 60

No of Tickets with PKCA-Kerberos system:

Group	User Requests	SGT Tickets
G1	[G1, S1] + [G1, S2]	30 + 0 = 30
G2	[G2, S1] + [G2, S2]	0 + 30 = 30

In case of PKCA-Kerberos user will get SGT only when the authorization & user context verification is successful. This means total tickets generated in this case will be limited to authorized users only. In this example there is 50% reduction in tickets compare to Kerberos V5. Below Fig. 6 shows the ticket comparison graph.

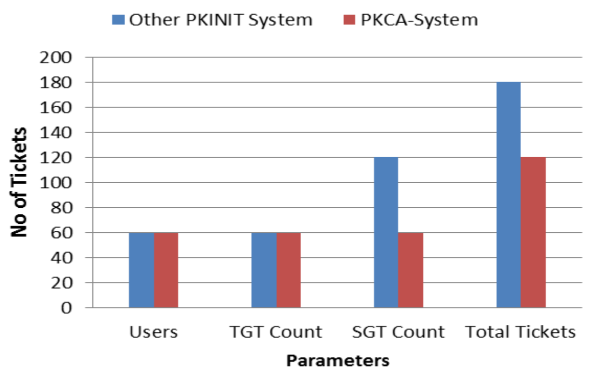

Fig. 6 Ticket Comparison Chart

Conclusion

We have introduced and analyzed PKCA-Kerberos system. To enhance security we have used public key authentication in kerberos phase-I which eliminates the risk associated with Password. This feature makes it more scalable and less vulnerable.

Looking at necessity of context aware authorization needs we have modified Kerberos in phase-II to restrict service tickets when user supplied context is not according to organization's policy. Our experiment shows that with enhancement Kerberos very well fulfills custom authorization needs.

The overall modification of our study is to add a new security layer of protection onto Kerberos authentication as well as authorization. Through this research we are making an effort to address authentication weaknesses and dynamic authorization problems related with mobile users using portable devices over unsecured network. This meets the flexibility and availability need of mobile user also address security needs of organizations before exposing services over internet

REFERENCES

[1] J. Kohl, C Neuman RFC: 1510: The Kerberos Network Authentication Service (V5), September1993.

[2] S. Hartman, K. Raeburn and C. Neuman. RFC: 4120: The Kerberos Network Authentication Service (V5), July 2005.

[3] Ravi Ganesan ,"Yaksha: Augmenting Kerberos with Public Key Cryptography", in: Network and Distributed System Security, 1995, Proceedings of the Symposium on, pp.132 143.

[4] Hu, Dahui ; Du, Zhiguo, "An Improved Kerberos Protocol Based on Fast RSA Algorithm", in Information Theory and Information Security (ICITIS),2010 IEEE International Conference , pp. 274 278.

[5] L. Zhu and B. Tung. RFC: 4556: Public Key Cryptography for Initial Authentication in Kerberos (PKINIT), Jun 2006.

[6] Downnard, I., "Public-key cryptography extensions into Kerberos", in IEEE Journals Volume: 21, Issue: 5, pp. 30 34, 2002.

[7] Hussein Khalid Abd-Alrazzaq,"Improvement Public Key Kerberos Using Identity-Based Signcryption ",in Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing, 2013, pp. 125-136.

[8] Dorothy E. Denning, Peter F. MacDoran, "Location based Authentication: Grounding Cyberspace for Better Security, Internet besieged: countering cyberspace sco²aws", Addison Wesley 1998, pp.167174.

[9] Abdelmajid, N.T., "Location-Based Kerberos Authentication Protocol", in social Computing (SocialCom), 2010 IEEE Second International Conference,pp. 1099- 1104.

[10] Sandhu, Ravi S.,"Role-based access control models", in Computer, 1996, Volume: 29, Issue: 2,pp. 38-47.

[11] Bertino,E., "Location-Aware Authentication and Access Control Concepts and Issues",in Advanced Inormation Networking and Applications, AINA '09.International Conference, pp. 10-15.

[12] Van Cleeff, A. "Bene¯ts of Location-Based Access Control: A Literature Study" in Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber,Physical and Social Computing (CPSCom), pp.739-746.

[13] Chris Amaris, Rand Morimoto, Microsoft System Center 2012 Unleashed, Pearson Education, Inc. pp.87-89.