Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Inserimento di nuove sintassi in LDAP per supportare gli Attribute Certificate

Giovanni Schmid – Luca Tamburo

RT-ICAR-NA-08-07 **dicembre 2008**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Inserimento di nuove sintassi in LDAP per supportare gli Attribute Certificate

Giovanni Schmid[1] – Luca Tamburo[1]

Rapporto Tecnico N.:
RT-ICAR-NA-08-07

Data:
dicembre 2008

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli

Inserimento di nuove sintassi in LDAP per supportare gli Attribute Certificate

G. Schmid L. Tamburo

30 novembre 2008

Abstract

X.509 è lo standard utilizzato in Internet sia per le funzionalità di autenticazione, tramite l'uso dei certificati a chiave pubblica (PKC), che per quella di autenticazione, mediante l'introduzione di un nuovo tipo di certificati, gli Attribute Certificate (AC).
Lightweight Directory Access Protocol (LDAP) d'altra parte è uno dei protocolli di directory service che negli ultimi anni ha suscitato sempre maggior interesse per le sue funzionalità; divenendo lo standard di fatto per tali servizi. Attualmente, LDAP fornisce un supporto per i PKC, ma non per gli AC.
Nel seguente lavoro, proponiamo un'estensione di LDAP per includere anche il supporto degli AC, presentandone un'implementazione per la suite opensource OpenLDAP. Inoltre, presentiamo un esempio d'uso di tale tecnologia per risolvere alcune problematiche legate agli ambienti collaborativi distribuiti di tipo grid.

Keywords

LDAP, X.509 Attribute Certificate, Privilege Management Infrastructure, Certificate Repository, Direct Delegation

1 Introduzione

Negli ultimi anni gli ambienti collaborativi distribuiti si sono imposti nello scenario dell'IT grazie alla progressiva diminuzione dei costi per la banda larga, il più importante ambiente di questo tipo per quanto concerne il calcolo ad alte prestazioni è sicuramente quello *grid*.
In tale ambito imprese ed enti mettono a disposizione le proprie risorse ad altri utenti della griglia, a questo punto si ha la neccessità di dover regolamentare l'accesso a tali risorse da parte degli utenti del sistema. Perchè l'ambiente sia veramente distribuito, non è quindi possibile avere una gestione degli utenti e delle risorse centralizzate.
LDAP, nato per sostituire DAP come protocollo di directory service, negli ultimi anni è stato oggetto di numerosi RFC, volte ad estenderne le funzionalità. Infatti, da directory service di tipo generico, si è passati ad utilizzarlo come information service, e di recente si è imposto come standard di fatto come repository per i certificati X.509; possiamo quindi affermare che LDAP sia lo standard di fatto per risolvere le problematiche presentate dagli ambienti distribuiti.
Lo standard per l'identificazione di un'entità è rappresentato dai *certificati a chiave pubblica* (PKC) e dalla sua infrastruttura la *Public Key Infrastructure* (PKI), entrambe definite in X.509. Nella quarta edizione di tale standard è stato introdotto un nuovo tipo di certificato, i *certificati degli attributi* (AC) e l'infrastruttura per il loro utilizzo *Privilage Managment Infrastructure* (PMI), come standard per le informazioni di autorizzazione. Tali infrastrutture presentano concetti tra loro simili.
Un confronto tra PKI e PMI è riportato in tabella 1. Un PKC, può essere visto come una specializzazione di un AC. L'entità che firma digitalmente un PKC è chiamata *Certification Authority* (CA) e l'entità che firma digitalmente un AC è chiamata *Attribute Authority* (AA). La radice delle fiducia nella PKI è a volte chiamata *root CA*, mentre la radice della fiducia nella PMI è chiamata *Source of Authority* (SOA). Le CA possono avere CA subordinate di cui si fidano e a cui delegano la possibilità di emettere PKC. Allo stesso modo le SOA, possono delegare i propri poteri di autorizzazione ad AA subordinate. Se un utente necessita che il suo PKC sia revocato, una CA emetterà un certificato di revoca (*Certificate Revocation List* CRL), allo stesso modo, se un utente necessita che i suoi privilegi siano revocati, una AA emetterà un certificato di revoca (*Attribute Certificate Revocation List* ACRL)[1].

Dato il forte legame che vi è tra la PKI e la PMI, e visto che LDAP supporta già la PKI, con il presente lavoro abbiamo voluto estendere LDAP in modo da supportare anche al PMI.
Il capitolo 2 espone il supporto degli AC in LDAP, nel capitolo 3 vengono esposti i passi necessari ad implementare un'operazione di ricerca in LDAP, nel capito 4 viene presentata la nostra implementazione in OpenLDAP, nel

Concetto	**PKI**	**PMI**
Certificato	Certificato a chive pubblica	Certificato degli attributi
Issuer del certificato	Certification Authority	Attribute Authority
Utente del certificato	Subject	Holder
Legame del certificato	Nome del Subject alla chiave pubblica	Nome dell'Holder agli attributi dei privilegi
Revoca	Certificate Revocation List (CRL)	Attribute Certificate Revocation List (ACRL)
Radice della fiducia	Root certification authority o trust anchor	Source of Authority
Autorità subordinate	Certification Authority subordinate	Attribute Authority

Tabella 1: Confronto tra PKI e PMI (tratta da [1])

capitolo 5 viene esposto il caso di studio della *delega diretta*, infine, nel capitolo 6 vengono esposte le conclusioni e gli sviluppi futuri.

2 Supporto degli AC in LDAP

La distribuzione degli Attribute Certificate può essere effettuata seguendo due modelli: il *pull* e il *push*.
Il primo modello è adatto nei casi in cui i privilegi di un'entità sono assegnati all'interno del suo dominio. In tal caso sarà l'entità stessa a presentare l'AC al server quando richiede di accedere ad una risorsa. Quindi il server non ha oneri di ricerca per quanto riguarda l'AC, e deve controllare tramite l'uso delle ACRL che tale certificato non sia stato revocato dall'AA (o dalla SOA) che lo ha emesso.
Il secondo modello, invece, è adatto nei casi in cui i privilegi sono assegnati all'interno del dominio del server, quindi sarà compito di quest'ultimo recuperare l'AC in un apposito *repository*.
Questo secondo metodo semplifica le operazioni che l'entità deve effettuare per accedere ad una risorsa [2].
Lo standard X.509 [3] prevede che gli AC e le ACRL (e le *delta-ACRL*) siano distribuite tramite servizi di directory; LDAP è attualmente lo standard di fatto per implementare i servizi di directory. Esso ha un'architettura distribuita, funziona sullo stack TCP-IP e può contenere tipi di dati eterogenei.
LDAP ha, quindi, sia il compito di conservare gli AC che un server può richiedere per prendere decisioni di autorizzazione, sia il compito di fornire informazioni sullo stato di un AC tramite le ACRL.
Un verificatore dei privilegi, deve essere in grado di interrogare un server

LDAP che funga da repository per gli AC, in modo da porter ricercare un AC avente i valori di ricerca in uno specifico campo dell'AC. Possiamo prendere ad esempio la matching rule *attributeCertificateExactMatch* definita in X.509 [3], in questo caso si ricerca un AC identificato dal suo *serialNumber* e dal suo *issuer* (questo è individuato tramite il *serialNumber* e il *subject* del PKC usato per firmare l'AC). Naturalmente, le necessità di ricerca non si limitano solo a questi due campi dell'AC. Ad esempio si può voler cercare un AC che abbia un determinato issuer e un determinato periodo di validità, tale ricerca può essere eseguita tramite la matching rule *attributeCertificateMatch*, sempre definita in X.509.
Per supportare tali funzionalità di ricerca, si potrebbe pensare di inserire, per ogni AC contenuto in LDAP, degli attributi che contengano i singoli campi dell'AC su cui effettuare la ricerca; tale soluzione però presenta alcuni inconvenienti. Primo, se una entry contiene più AC, questi non si possono distinguere tramite gli attributi estratti, in quanto gli attributi di una entry LDAP sono un insieme di valori e non una sequenza di valori. Per risolvere tale problema vi deve essere una entry per ogni AC, tali entry dovrebbero essere inserite come figlie della entry che rappresenta l'*holder* dell'AC [4]. In questo caso le entry di un repository aumenterebbero in modo notevole, infatti, sono necessarie tante entry quanti sono gli AC emessi per un determinato holder, oltre la entry che rappresenta l'holder stesso; una situazione del genere si verifica spesso visto che un singolo holder può avere numerosi AC emessi da differenti AA e per differenti usi. In secondo luogo, la ricerca non viene effettuata sull'AC ma sugli attributi da esso estratti. Dato che gli attributi di una entry LDAP possono essere modificati e non vi è alcun modo per assicurare la loro integrità, non si può essere certi che i valori degli attributi estratti corrispondano con l'AC a loro associato. Il problema dell'integrità invece non si presenta per gli AC data la loro struttura [2]. Sia la prima, sia la seconda osservazione, sono state già effettuate in merito ai PKC in [5].
Gli AC sono codificati come struttere ASN.1, ogni campo dell'AC, può essere un tipo semplice, ad esempio un INTEGER o una BIT STRING, o può essere un tipo strutturato composto da più campi semplici o anch'essi strutturati.
In [6] si fornisce un metodo per effettuare confronti tra un valore e un componente specifico di un attributo. Tale metodo si basa sui tipi ASN.1, definendo come ci si debba riferire ad un componente all'interno di un valore di un attributo e come confrontare il componente riferito con un valore di asserzione. Le regole di confronto sono definite sia per i tipi base sia per i tipi complessi ASN.1; vengono anche definite una nuova asserzione e un nuovo filtro su misura per ogni componente. Il filtro di ricerca per il component matching è una asserzione di una regola di confronto la cui regola di confronto è *componentFilterMatch*, il cui valore di asserzione è un component filter. Il filtro di ricerca per il component matching è costituito da tre

parti:

- **Component Reference**: specifica quale componente del valore dell'attributo sarà confrontato con il valore di asserzione.
- **Matching Rule**: specifica quale regola di confronto sarà usata per effettuare il confronto sui valori.
- **Value**: un valore di asserzione codificato in GSER[1].

Il metodo appena descritto, se confrontato con quello precedentemente esposto, presenta i seguenti vantaggi [5]:

1. Non si devono estrarre i componenti dell'AC e conservarli separatemante da esso, evitando in tal modo una duplicazione dei dati e problemi di integrità tra l'AC e gli attributi che rappresentano i componenti da esso estratti.
2. Il confronto non viene eseguito tra il contenuto degli attributi che rappresentano i componenti estratti, ma viene eseguito sull'AC stesso. In tal modo si evita la restituzione di più AC se la entry ne contiene più d'uno, ma si restituisce solo l'AC corrispondente ai valori di ricerca.
3. Tale metodo diviene conveniente per fornire un meccanismo di confronto complesso e flessibile perchè il confronto tra gli attributi e i valori di ricerca viene effettuato al livello ASN.1.

Uno dei principali problemi del supporto in LDAP per i certificati X.509 deriva dal fatto che la codifica nativa di LDAP è composta o da una stringa binaria o da una stringa di caretteri ASCII. Con questa codifica è particolarmente difficile riprodurre l'informazione strutturata di un tipo ASN.1. Per risolvere tale problema in [7] è definita la codifica GSER (Generic String Encoding Rules). GSER viene usata come codifica base per i valori di ricerca delle regole di match. Tale codifica genera una stringa di caratteri UTF-8 per rappresentare un tipo ASN.1 tramite un insieme predeterminato di caratteri, riportati in tabella 2, per mantenere la struttura originaria di tipo ASN.1.
GSER definisce al livello più basso una codifica dei tipi base ASN.1, quali INTEGER, STRING e BOOLEAN e a un livello più alto una codifica dei tipi ASN.1 più complessi, come SEQUENCE o SET.
In [8] sono riportate le regole di match definite per gli AC in X.509 [3] in codifica GSER.

[1] Vedi seguito

Simbolo	Significato
{ }	delimitano una sequenza
sp	zero o più caretteri spazio
mps	uno o più caretteri spazio
sep	separatore rappresentato dal carattere “,”
dquote	rappresenta il doppio apice

Tabella 2: Alcuni dei caratteri utilizzati dalla codifica GSER

3 Ricerca degli AC in LDAP

Nel seguito indentificheremo rispettivamente con **C** un client che effettua un'operazione di ricerca in LDAP, **S** il server LDAP, i_C l'identità del client, f_C il filtro di ricerca codificato tramite GSER e r_S la risposta del server LDAP.
Il client **C** vuole effettuare una ricerca in un repository LDAP, invia un messaggio del tipo

$$\textbf{a) C} \longrightarrow \textbf{S: } (i_C, f_C);$$

dichiarando la propria identità (i_C) ed inviando un filtro di ricerca (f_C).
Quando il server **S** riceve il messaggio di tipo **a)** elabora tale richiesta nel seguente modo:

1. Preleva f_C, e controlla che la sua sintassi sia valida. Se la sintassi non è valida, il server termina la ricerca inviando un messaggio del tipo

   $$\textbf{b) S} \longrightarrow \textbf{C: } r_S,$$

   inserendo in r_S un codice di errore.

2. Se invece, la sintassi di f_C risulta valida, il server lo elabora, estrando da esso i valori dei componenti coinvolti nella ricerca, con tali valori costriusce poi il filtro normalizzato ($\bar{f}_C$), caratterizzato dal minor numero possibile di caratteri spazio, cosi come richiesto dalla grammatica che specifica il filtro di ricerca [8].

3. A questo punto il server legge un AC dal suo repository, da esso vengono estratti i valori dei componenti coinvolti nella ricerca e con essi viene costruito un nuovo filtro normalizzato $\bar{f}_S$.

4. Il server confronta $\bar{f}_C$ con $\bar{f}_S$, se il confronto ha esito positivo, il server invia un messaggio di tipo **b)** al client inserendo l'AC in r_S. Se invece, il confronto ha esito negativo, il server ripete il passo precedente leggendo un nuovo certificato.

Se il server non trova nessun certificato che corrisponde al filtro $\bar{f}_S$ esso invierà al client un messaggio di tipo **b)** e r_S conterrà un codice di errore.

4 Implementazione in OpenLDAP

L'implementazione da noi realizzata consiste nell'inserimento della sintassi **AttributeCertificate** [3, capitolo 12]. Tale sintassi risulta essere fondamentale, in quanto rappresenta la struttura di un AC. Abbiamo poi proceduto all'inserimento della sintassi di ricerca **AttributeCertificateExactAssertion** [3, paragrafo 17.3.1], a cui corrisponde la matching rule **attributeCertificateExactMatch** già definita in §2.
Tali modifiche hanno riguardato solo la parte server di OpenLDAP (*slapd*), non è stato necessario apportare nessuna modifica al client.
Per quanto concerne la sintassi di un AC, è stata implementata la funzione `attributeCertificateValidate`, che si occupa di controllare che un valore inserito in LDAP di tipo AC abbia una struttura conforme a quanto specificato da [3]. Tale funzione utilizza le API di codifica DER/BER (Distinguished Encoding Rules/Basic Encoding Rules[2]) che OpenLDAP già utilizza per i PKC; tali API sono documentate in [10].
Per quanto concerne la sintassi di ricerca si è utilizzata la grammatica ABNF fornita in [8], da cui è stato elaborato il seguente filtro GSER, dove, _sn_ rappresenta il serial number dell'AC, mentre _i_dn_ e _i_sn_ rappresentano il distinguished name e il serial number del PKC usato dall'issuer per emettere l'AC. Per il valore degli altri simboli si faccia riferimento alla tabella 2 e a [7].

```
{ sp serialNumber mps _sn_ sep sp issuer mps { sp baseCertificateID
mps { sp issuer mps { sp directoryName:rdnSequence:dquote
_i_dn_ dquote sp } sep sp serial mps _i_sn_ sp } sp } sp }
```

Per realizzare i passi esposti nel §3 rispetto alla sintassi di ricerca *attributeCertificateExactAssertion* sono state implementate le seguenti funzioni, il cui prototipo è specificato in tabella 3:

- `serialNumberAndIssuerSerialValidate`: si occupa di validare il filtro di ricerca.
- `serialNumberAndIssuerSerialCheck`: si occupa di estrarre dal filtro di ricerca i valori delle componenti.
- `serialNumberAndIssuerSerialPretty`: si occupa di creare un filtro di ricerca normalizzato.
- `attributeCertificateExactNormalize`: si occupa di creare un filtro normalizzato dato un AC.

[2]Tali codifiche sono definite dallo standard [9]

- `serialNumberAndIssuerSerialNormalize`: si occupa di creare un filtro normalizzato se l'input della funzione `attributeCertificateExactNormalize` è un filtro invece che un AC.

```
static int serialNumberAndIssuerSerialValidate
 ( Syntax *syntax, struct berval *in )
```

```
static int serialNumberAndIssuerSerialCheck
 ( struct berval *in, struct berval *sn, struct berval *is,
 struct berval *i_sn, void *ctx )
```

```
int serialNumberAndIssuerSerialPretty
 ( Syntax *syntax, struct berval *in, struct berval *out, void *ctx )
```

```
static int attributeCertificateExactNormalize
 ( slap_mask_t usage, Syntax *syntax, MatchingRule *mr,
 struct berval *val, struct berval *normalized, void *ctx )
```

```
static int serialNumberAndIssuerSerialNormalize
 ( slap_mask_t usage, Syntax *syntax, MatchingRule *mr,
 struct berval *in, struct berval *out, void *ctx )
```

Tabella 3: Prototipi delle funzioni per il supporto della PMI in OpenLDAP

5 Caso di studio: Delega diretta

Un esempio applicativo delle estensioni da noi introdotte può essere rappresentato dalla *delega diretta*, una funzionalità del controllo degli accessi, che consente ad un utente (*guest*) di allocarsi su di un host come *ospite* di un account standard (*sponsor*) di tale host.
Per realizzare un ambiente di delega diretta abbiamo creato un repository LDAP per rappresentare le utenze, possiamo quindi, suddividere tali utenze in tre gruppi: *utenti ordinari*, *utenti sponsor* e *utenti guest*. Esaminiamo ora le ultime due categorie. Gli utenti sponsor sono utenti ordinari a cui è stato concesso il ruolo di *amministratori delle proprie risorse* in riferimento ad altri utenti ordinari del sistema, ma anche verso utenti che non hanno un account sul sistema. Gli utenti guest sono utenti che possono avere o meno un account sul sistema, ma che accedono ad esso tramite un utente sponsor [11].
È possibile collegare tali utenti con le entità della PMI: gli utenti sponsor agiscono come AA, gli utenti guest agiscono come le entità finali, mentre l'amministratore del sistema assume il ruolo della SOA.
La figura 1 rappresenta il repository LDAP da noi creato, le categorie appena descritte sono raggruppate sotto il nodo **People**.
Per realizzare tale struttura sono state usate le *objectClass* definite negli schemi: *core*[3], *cosine* [12], *inetOrgPerson* [13] e *nis* [14]; inoltre abbiamo definito nello schema *guest.schema* la objectClass che rappresenta i nodi inseriti sotto **ou=People,ou=Guest**. Ogni nodo rappresentante un utente contiene le informazioni ad esso associate in un sistema unix-like oltre ai suoi PKC e AC.

[3]Tale schema utilizza varie RFC, per maggiori informazioni sulla sua definizione si consiglia la lettura del file schema implementato in OpenLDAP

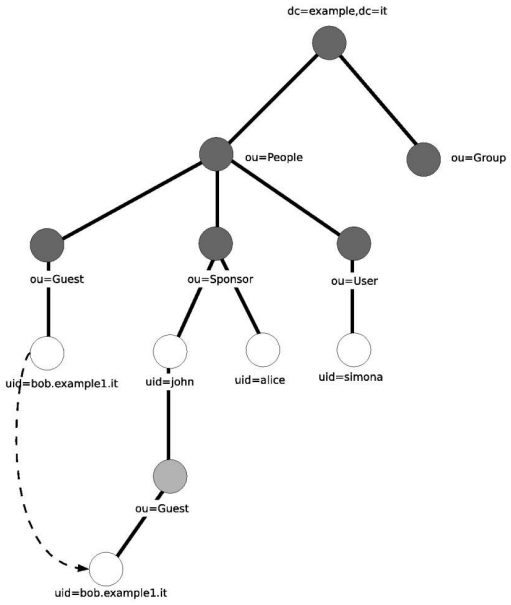

Figura 1: Repository LDAP per la delega diretta

Di seguito riportiamo un esempio che mostra l'inserimento di un AC e la sua ricerca.
Per inserire un AC creiamo un file LDIF[4] chiamato *newac.ldif* con il seguente contenuto (il valore attributeCertificateAttribute[5] è troncato per brevità).

```
dn: uid=bob.example1.it,cn=Delegate,uid=john,ou=Sponsor,ou=People,
 dc=example,dc=it
changetype: modify
add: attributeCertificateAttribute;binary
attributeCertificateAttribute;binary:: MIICLjCCAZcCAQEwaKBmMGGkXzB
 dMQswCQYDVQQGEwJJVDENMAsGA1UECBMEUm9tYTEP.........
```

A questo punto possiamo inserire l'AC dell'utente sponsor *bob.example1.it* lanciando il seguente comando.

```
ldapmodify -x -D "uid=john,ou=Sponsor,ou=People,dc=example,dc=it"
-W -f newac.ldif
```

Per controllare che tale inserimento sia andato a buon fine, effettuiamo la ricerca dell'AC appena inserito utilizzando la regola di ricerca **attributeCertificateExactMatch**, lanciamo quindi il comando

```
ldapsearch -x -LLL -b "dc=example,dc=it"
 "(attributeCertificateAttribute:attributeCertificateExactMatch:={
```

[4] *LDAP Data Interchange Format* definito in [15].
[5] L'AC nel file LDIF è espresso tramite la codifica *Base-64* definita in [16].

```
serialNumber 0, issuer { baseCertificateID { issuer {
directoryName:rdnSequence:\"email=john@example.it,cn=john,
o=example,st=Napoli,c=it\" }, serial 16 } } })"
attributeCertificateAttribute
```

che ci restituisce il seguente output, troncato per brevità.

```
dn: uid=bob.example1.it,cn=Guest,uid=john,ou=Sponsor,ou=People,
 dc=example,dc=it
attributeCertificateAttribute;binary:: MIICLjCCAZcCAQEwaKBmMGGkXzB
 dMQswCQYDVQQGEwJJVDENMAsGA1UECBMEUm9tYTEP.........
```

6 Conclusioni e sviluppi futuri

L'obiettivo di questo lavoro è stato quello di includere in LDAP il supporto per gli AC e realizzarne un'implementazione per OpenLDAP. Tale obiettivo è stato raggiunto tramite l'inserimento della sintassi **AttributeCertificate** [3, capitolo 12] e della sintassi di ricerca **AttributeCertificateExactAssertion** [3, paragrafo 17.3.1] a cui corrisponde la regola di match **AttributeCertificateExactMatch**. Tali modifiche ad OpenLDAP sono state sottomesse alla comunity, e sono attualmente in fase di testing[6].
Come già detto in §2, le necessità di ricerca non si limitano solo alla regola di match precedente; per un supporto completo andrebbero implementate le regole di match e le sintassi definite in [3, capitolo 17]. Tale implementazione sarà oggetto di un prossimo lavoro.
Per effettuare il testing delle funzionalità qui presentate è stato implementato un software per la creazione degli AC, tale software presenta però limitazioni molto pesanti e non si presta ad un utilizzo reale da parte di utenti finali. Per tali motivi, sarebbe opportuno estendere un software opensource come OpenSSL, rendendolo in grado di supportare gli AC nello stesso modo in cui attualmente supporta i PKC.

Riferimenti bibliografici

[1] D.W. Chadwick. An X.509 role based privilege management infrastructure. In *Briefing - Global InfoSecurity 2002, World Markets Research Centre Ltd.* World Markets Research Centre, October 2001. On accompanying CD-ROM Reference Library/03.pdf.

[2] S. Farrell and R. Housley. *An Internet Attribute Certificate Profile for Authorization RFC3281.* April 2002.

[3] International Telecommunication Union Telecommunication Standardization Sector. *The Directory: Public-key and attribute certificate frameworks.* ITU-T, March 2000.

[6] OpenLDAP ITS – `http://www.openldap.org/its/index.cgi?findid=5695`

[4] D. W. Chadwick and M. V. Sahalayev. *Internet X.509 Public Key Infrastructure LDAP Schema for X.509 Attribute Certificates darft–ietf–pkix–ldap–ac–schema–02.txt*. October 2004.

[5] S. S. Lim, J. H. Choi, and K. D. Zeilenga. *Design and Implementation of LDAP Component Matching for Flexible and Secure Certificate Access in PKI*. April 2005.

[6] S. Legg. *Lightweight Directory Access Protocol (LDAP) and X.500 Component Matching Rules. RFC3687*. February 2004.

[7] S. Legg. *Generic String Encoding Rules (GSER) for ASN.1 Types RFC3641*. October 2003.

[8] D. W. Chadwick and S. Legg. *Internet X.509 Public Key Infrastructure LDAP Schema and Syntaxes for PMIs draft–ietf–pkix–ldap–pmi–schema–00.txt*. June 2002.

[9] International Telecommunication Union Telecommunication Standardization Sector. *ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*. ITU-T, July 2002.

[10] *Man page (section 3): lber-decode, lber-encode, lber-memory, lber-sockbuf, lber-types*.

[11] G. Laccetti and G. Schmid. A PMI-aware extension for the SSH service. In *Parallel Processing and Applied Mathematics*, volume 4967 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007.

[12] K. Zeilenga. *COSINE LDAP/X.500 Schema RFC4524*. June 2006.

[13] M. Smith. *Definition of the inetOrgPerson LDAP Object Class RFC2798*. April 2000.

[14] L. Howard. *An Approach for Using LDAP as a Network Information Service RFC2307*. March 1998.

[15] G. Good. *The LDAP Data Interchange Format (LDIF) - Technical Specification RFC2849*. June 2000.

[16] N. Freed and N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies RFC2045*. November 1996.