An Identity-Based Security Infrastructure for Cloud Environments

Christian Schridde[1], Tim Dörnemann[1], Ernst Juhnke[1], Bernd Freisleben[1], Matthew Smith[2]

[1]Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str. 6, D-35032 Marburg, Germany
Email: {schriddc, doernemt, ejuhnke, freisleb}@informatik.uni-marburg.de

[2]Distributed Computing Security Group, University of Hannover
Schlosswenderstr. 5, D-30167 Hannover, Germany
Email: smith@dcsec.uni-hannover.de

***Abstract*— This paper presents a novel security infrastructure for deploying and using service-oriented Cloud applications securely without having to face the complexity associated with certificate management. The proposal is based on an identity-based cryptographic approach that offers an independent setup of security domains and does not require a trust hierarchy compared to other identity-based cryptographic systems. The service URLs can be used as public keys, such that creating a secure connection to a service is very simple. A comparison between traditional approaches and identity-based cryptography with respect to data transfer requirements is presented.**

I. Introduction

The Grid computing paradigm offers unified access to heterogeneous compute and storage resources in a shared use environment. To facilitate Grid usage, a large number of supporting services, such as authentication, authorization, file staging, virtual organization management, and (meta-) scheduling, are offered by the widespread Globus GT4 middleware or by software systems, such as the GridWay meta-scheduler or VOMS for virtual organization management. One of the main points of critique leveled against the Grid is the complexity of installing and securing custom applications. Much of this complexity stems from the multi-administrator, shared use, heterogeneous nature of the Grid and the highly diverse requirements Grid users have. For a detailed discussion of these problems, the reader is referred to Smith et al. [12].

The emerging Cloud computing paradigm, as pioneered by Amazon, offers simple and quick access to compute and storage resources offered by a single provider. Users may access resources on-demand and in a pay-per-use manner through simple interfaces like web services, without knowledge or control of the technology and the infrastructure used by the provider. Application areas include: application hosting, backup and storage, content delivery, and computing resources. These application areas differ from Grid computing where the main focus lies on high performance computing and data storage. One of the major advances of Cloud computing over Grid computing is the use of a fully virtualized infrastructure, which significantly eases the installation and maintenance of custom applications. This also solves many of the shared use security issues previously faced by Grid infrastructures [14]. However, currently Cloud computing offers little in the way of the support infrastructure that has been established in the Grid, instead it simply offers access to the raw resources the user must use manually. For simple applications, this is not much of a problem, but many of the complex service-oriented applications currently running in the Grid rely on the management and security functionality offered by the Grid middleware and the supporting products. However, it is not trivial to install and use a Grid middleware such as GT4 in a Cloud, since the Grid Security Infrastructure (GSI) is incompatible with both the virtual Cloud environment and the usage scenario.

Grid security systems typically offer a great number of features, not all of which are commonly used, but significantly hinder quick and easy setup and usage. Furthermore, the virtual, transient nature of Cloud resources would create problems for the host name based certificates used by GSI, hindering an adoption of GSI for Cloud computing.

In this paper, we introduce a security infrastructure for service-oriented Cloud applications that overcomes the problems of previous Grid security solutions and allows complex applications to be deployed and used on-demand in a secure manner. Our proposal makes use of a novel identity-based cryptographic (IBC) system to avoid the complexity and management problems of certificate-based security infrastructures. With an IBC approach, any string can be used as a public key. This leads to a much more intuitive security environment without the need for a heavyweight certificate infrastructure. Previously, IBC systems have not seen widespread adoption, since IBC systems require a single security provider or a hierarchy of cooperating security providers with a single root of trust to set up the system. Not only is it unlikely that all Cloud resource providers will want to cooperate or have a single root of trust, but this drawback of IBC systems creates a significant management overhead that defeats the benefit of easy usage promised by IBC. In contrast, the proposed novel IBC security system allows multiple security domains to operate independently of each other. Thus, one of the main

978-1-4244-5849-3/10/$26.00 ©2010 IEEE

problems of IBC systems is solved, and the system is very useful for a complex environment like service-oriented Cloud computing.

The paper is organized as follows: In Section II, the problem statement is presented. Section III discusses related work. Section IV introduces the proposed approach to use identity-based encryption to overcome the problems. Section V presents a security overhead comparison between identity-based cryptography and certificates. Section VI concludes the paper and outlines areas for future research.

II. Problem Statement

Common web service security standards, such as WS-Security and WS-SecureConversation, provide features for signing and encrypting messages that are exchanged between web services and clients. Another possibility to secure the communication between web services is to encrypt or sign the entire data stream using transport layer security (TLS).

In WS-SecureConversation, the client establishes a context with the server with the initial message. This context has the purpose of authenticating the client identity to the server and of establishing a shared secret key. As soon as the context is established, messages can be secured (signed or encrypted) using the shared secret key from the context. For the initial key exchange, WS-Security and WS-Trust are used. To assure that the server's public key is not altered during transfer, it is signed with the server's X.509 certificate. The WS-SecureMessage approach is simpler, since the client does not establish a context before sending data (i.e., invoking operations) to the server. It uses public key mechanisms and utilizes existing keys, such as the server's public key from its certificate.

TLS also makes use of certificates to sign the public keys before exchanging them. A public key is needed to encrypt the chosen symmetric key before transferring it to the communication partner.

In the area of Grid computing, the mentioned standards have been implemented in GSI that supports authentication through X.509 certificates and delegation through proxy certificates and the WS-Trust standard. Delegation allows a client to delegate a proxy certificate to a service. The target service can then perform tasks on behalf of the user who owns the proxy certificate. A proxy is derived from the user's certificate and consists of a new certificate and a private key. The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. It is signed by the owner, rather than a Certificate Authority (CA). Proxies have limited lifetimes, i.e. the proxy should no longer be accepted by others after the lifetime has expired.

If services from different providers are to be invoked, one might have to deal with different trust domains, meaning that one needs to trust more than one certificate authority, and different user certificates might be needed. This makes the invocation of services quite complicated. In particular, the composition of secure workflows, for instance using the Business Process Execution Language (BPEL) [5], becomes unmanageable, since typically many services (from different trust domains) are to be invoked during the execution of a workflow.

In Figure 1, a sample workflow from the area of multimedia analysis is shown. The workflow receives a – potentially large – video file including an audio stream. At first, face and speech detection algorithms are applied in parallel. Face and speaker recognition are applied if the preceding workflow steps produced positive results (the diamond symbols indicate conditional transitions). The resulting meta-information is then used as the input for actor recognition. In parallel, cut and text detection are performed. Optical character recognition (OCR) based on the latter and speech recognition produce the input for the parental guidance advisor. Together with the generated keyframes, all extracted information are stored using the MovieDB service. As this simplified workflow demonstrates, the analysis of a video requires the invocation of a large number of partner services. It is not clear when and if a service (for instance *FaceRecognition*) is executed. Furthermore, these services might be installed on different hosts and, if provided by different companies, would run in different trust domains.

Since different users are able to deploy services on a Grid site, even within a single site, different trust domains may exist. If, for instance, company A and company B are competitors and deploy services on the same Grid site, their services should be isolated from each other, such that private information (intellectual property) remains private. Consequently, security per service is necessary, rather than security per host. This issue has already been addressed on the code level by Smith et. al [13]. However, if the communication is secured using the same host certificate, then key agreement could be eavesdropped. As a consequence, using the traditional certificate-based approach, a separate certificate for every service, or at least every trust domain within the Grid, would be necessary. This would further increase the number of certificates to be managed.

To speed up the sample workflow from above, Cloud resources could be utilized to execute the parallel branches of the workflow in parallel on different Cloud machines. If non-dedicated resources, for instance on-demand resources from Cloud infrastructures like Amazon's Elastic Compute Cloud (EC2) [1], should be utilized instead or in addition to dedicated resources, further problems arise.

What makes the use of certificates delicate is that prior to the provisioning of the on-demand resources, the host names of the provided resources are unknown to the requestor. The host names are determined at startup of the machines, meaning that one would have to create and sign a host certificate on-the-fly every time a resource is acquired. Since self-signing certificates is not an option, this procedure is time-consuming and cost-intensive. Even if certificates are signed by a well-known, trusted CA, security attacks are possible due to the high dynamics of such infrastructures.

In Figure 2, all necessary steps to securely invoke a service using certificates on an on-demand resource are displayed. After the user starts the BPEL workflow (1), the BPEL engine has to request the (first) target host's certificate (2, 3). Then,

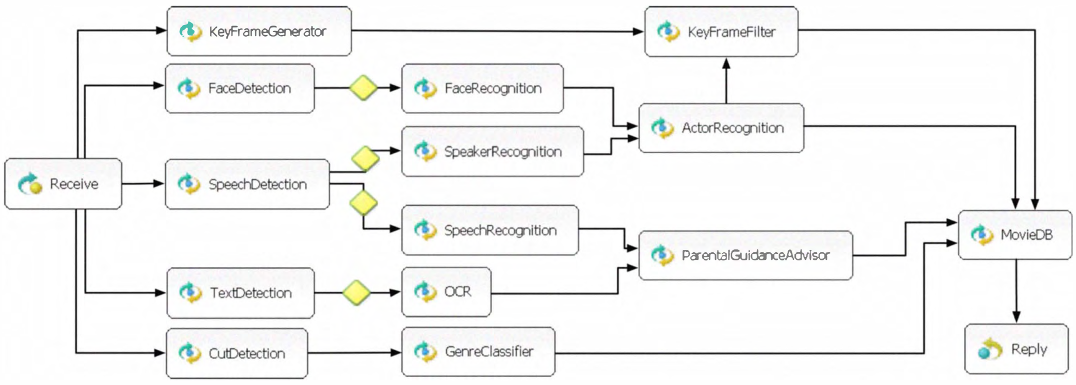

Fig. 1. Sample multimedia analysis workflow

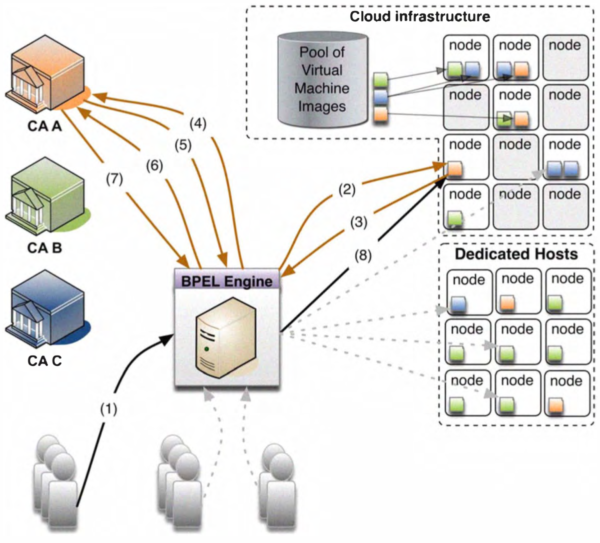

Fig. 2. Steps required for a service invocation using certificates

it must retrieve the corresponding root certificate (4, 5), if not yet present. After verifying the validity of the host certificate, it must be checked whether the certificate has been revoked or not (6, 7). In step (8), the actual call to the target service can be executed. This procedure has to be repeated for every target host in the workflow.

Certificates have a fixed lifetime that may be longer than the time the on-demand resource (i.e., the host name) belongs to the user (A) or organization the certificate is issued to. If then another user (B) requests a resource that coincidentally is assigned to the host name previously used by user A, two valid certificates for the machine exist. The two certificates are then owned by entities that are likely to be in different trust domains. As a result, user A may perform a man-in-the-middle-attack (MITMA) against B. If B wants to access the requested resource, A could intercept the communication (step (2) and the corresponding response (3) in Figure 2) and – using the valid certificate for the host name – alter or steal exchanged data. Due to the high number of service calls and involved partners in typical workflows, this scenario is quite likely to occur. Tests with Amazon EC2 show that requested machines are often assigned to host names that have been assigned before. This security flaw could be circumvented by using revocation lists. Whenever a resource is deprovisioned, the certificate would have to be added to the list. Due to the highly volatile nature of such infrastructures, the list would grow dramatically.

III. Related Work

To the best of our knowledge, there are only a few attempts to apply IBC to Grid/Cloud computing. For example, Lim and Paterson [8] propose the use of identity-based cryptography as an alternative for GSI. They describe several scenarios in which IBC simplifies the current Grid security solutions, like the elimination of certificate chains, simple proxy generation, easy revocation of proxy certificates and the savings of bandwidth by using the pairing based approach proposed by Boneh and Franklin [2]. In the same way, Li et. al [7] propose to use IBC as an alternative to the SSL authentication protocol in a Cloud environment. However, like most other IBC systems, they suffer from the fact that a trust hierarchy must be created and the providers must cooperate to create a working system. This has hindered the adoption of this system in Grid computing and rules it out for Cloud computing, since many trust domains are involved. The same applies to the work of Crampton et al. [4] who examine how identity-based cryptography can be used to secure Web services in general. Lim and Robshaw [9] propose a Dynamic Key Infrastructure for Grids based on identity-based encryption. Furthermore, Chen et al. [3] discuss the use of a combination of short time private keys and long-term private keys as an alternative for GSI. None of the papers addresses the new paradigm of Cloud computing nor do they identify the need of per-service security and the arising danger of host-reuse MITMAs.

IV. A Novel IBC based Cloud Security Infrastructure

The use of IBC as the basis of a security infrastructure is beneficial whenever in a dynamic environment many parties want to communicate with each other and the communications endpoint addresses are already known or a secure naming service exists like DNSsec or the EC2 API, since in this case the PKI or certificate infrastructure is not needed. The workflow scenario described in Section II, where certificates are cumbersome to use and are vulnerable to MITMAs due to resource re-use and the lack of instant revocation capabilities, is ideal for an IBC based solution. The environment is highly dynamic, since in the used workflow it is not predetermined which services are called, at what times, and on which hosts, since this depends on the input data. Furthermore, if services invoke other services directly (as in workflow choreography), certificate-based approaches become even more cumbersome since potentially each service could call all other services, such that $N \times N$ certificates need to be exchanged and more critically kept up-to-date. Since all involved services must have a unique communication endpoint that must be known to the remote caller to execute the call, IBC offers a simplification of the security infrastructure compared to a certificate-based approach, as long as the used IBC does not require complex trust hierarchies to be created to operate. In previous work [10], we introduced an IBC approach that allows completely independent security providers to setup their security domain but still allows users of the IBC system to cross domain boundaries. This is a significant advantage of our system, since it really simplifies setup and use of the IBC approach, compared to the traditional IBC solutions that simply shift the administrative burden around and do not save much in effort or complexity compared to certificate-based systems. In this paper, the issues in adopting our IBC system to Cloud computing are discussed. A brief introduction to our IBC approach is given in the next subsection. For a more detailed mathematical discussion of our IBC proposal, the reader is referred to our previous paper [10]. Security proofs of the system are presented in [6].

A. Identity-based Cryptography

In 1984, Shamir published a paper [11] in which he proposed an idea to overcome the problem of the missing native binding between a public key and its owner. His new approach was called *Identity-based Cryptography* (IBC). The idea is to use the identity of the participant as the public key rather than a specially crafted integer. To send an e-mail to a business partner, it is sufficient to know the e-mail address to encrypt the entire message rather then grabbing the corresponding public key file from a public ring-server. IBC simplifies key handling and human verification of a public key to visual checking of a readable string. IBC solves no problem which can not also be solved with public key infrastructures and CA s, but significantly reduces the complexity of a security system. To use an arbitrary string as a public key, a private counterpart that is intractable to compute without knowing a special secret must exist. The generation of these private keys is performed by identity-based private key generators ($\mathsf{ID}-\mathsf{PKG}$). Informally, an IBC system works as follows: Let ID be the identity string of a participant E_{ID}. Using a trapdoor one-way function, an $\mathsf{ID}-\mathsf{PKG}$ extracts a private key from the identity string ID using a secret trapdoor that is only known to the $\mathsf{ID}-\mathsf{PKG}$. Another participant is then able to encrypt a message using only the string ID (which is converted to an integer using a hash-function) and a public one-way function that can be trapdoored by the sender by using the private key of ID. There are two requirements that must be fulfilled such that IBC can be applied: the identities must be unique and they must be publicly known, especially to the caller.

B. SSF: Secure Session Framework

Our identity-based key agreement scheme, called SSF, is described briefly below.

1) Public, Shared Parameters: The public, shared parameters (PSP) of SSF is a quadruple (N, R, G, H) where:

- $N = PQ, P, Q \in \mathbb{P}$
- $R \in_R \mathbb{Z}_N$ and $\gcd(R, (P-1)(Q-1))$
- $G \in_R \mathbb{Z}_N$ and $\mathrm{ord}_N(G) \gg 1$
- $H : \{0,1\}^n \to \mathbb{Z}_N$ a collision-free hash-function

2) Extracting the Identity Key: The $\mathsf{ID}-\mathsf{PKG}$ extracts the identity key d_{ID} out of ID by

$$H(\mathsf{ID})^{1/R} \equiv d_{\mathsf{ID}} \pmod N$$

Note, that computing the R-th root within $\mathbb{Z}_N$ can only be done when P, Q is known.

3) Building the Session Initiation Key: A participant E_{ID} uses the PSP to generate its Session Initiation Key (Sik) via

$$r \in_R \mathbb{Z}_N;\ \ \mathrm{Sik}(\mathsf{ID}, \mathsf{r}) = g^r d_{\mathsf{ID}} \pmod N$$

4) Key Agreement: The actual key agreement between two participants $\mathsf{E}_{\mathsf{ID}_\mathsf{A}}$ and $\mathsf{E}_{\mathsf{ID}_\mathsf{B}}$, after they have exchanged their Session Initiation Keys, is:

$$\mathsf{E}_{\mathsf{ID}_\mathsf{A}} : \left(\mathrm{Sik}(\mathsf{ID}_\mathsf{B}, \mathsf{r}_\mathsf{B})^R \cdot H(\mathsf{ID}_\mathsf{B})^{-1}\right)^{r_A} \equiv S \pmod N$$

$$\mathsf{E}_{\mathsf{ID}_\mathsf{B}} : \left(\mathrm{Sik}(\mathsf{ID}_\mathsf{A}, \mathsf{r}_\mathsf{A})^R \cdot H(\mathsf{ID}_\mathsf{A})^{-1}\right)^{r_B} \equiv S \pmod N$$

The integer S is the common session key that can be used to derive a secret key.

5) Signing and Verification: To prove the legal possession of an identity, a signature algorithm is used. Let m be a message to be signed by a private identity key, then the corresponding signature/verification steps are

$$\mathsf{Signing} : \mathrm{Sig}(m, \mathsf{r}, \mathsf{ID}) = \left(\mathrm{Sik}(\mathsf{ID}, H(m)r), m, G^{Rr}\right)$$

$$\mathsf{Verification} : \mathrm{Sik}(\mathsf{ID}, H(m)r)^R H(\mathsf{ID})^{-1} \overset{?}{\equiv} G^{RrH(m)}$$

If the last congruence is true modulo N, then the signature is valid.

C. IBC for Cloud Computing

Before a user is able to use Cloud resources, (s)he has to register itself at the corresponding authority server, for instance, the Amazon EC2 service. The registration includes specifying how to pay for the desired service and retrieving login credentials for the corresponding account. The authority keeps a database of already registered users and thus issuing two accounts with the same user name can be easily prevented.

During registration, each user also obtains a private identity key for the chosen account name. With this key, the user is able to prove that (s)he legitimately possesses the account. This can be achieved by signing a current time stamp or a random nonce. After the registration process is completed, the user can start to deploy his or her own services.

To achieve secure communication, the user has to assure that each of the self-deployed services also retrieves a private identity key corresponding to its service URL. For this purpose, the service must contact the $\mathsf{ID-PKG}$ in a secure way, such that the private key is not revealed to any malicious entity within the network. Most traditional IBC systems must fall back on protocols like SSL, TLS and IPSec for the distribution of the private identity key, thus creating further complexity if these have not been set up already. With our IBC system, the public parameters of the security domain can also be used to securely contact the $\mathsf{ID-PKG}$, since the $\mathsf{ID-PKG}$ has access to the secret parameters of the security domain, the primes P and Q. Since each service must know the PSP anyway, the moduli $N = PQ$, which is part of the public shared parameters, and a fixed public integer e can be used to contact the $\mathsf{ID-PKG}$ using a simple RSA encryption without the need for a further security infrastructure.

After a secured channel between the service and the $\mathsf{ID-PKG}$ has been established, the service specifies the URL it needs a private key for. To prevent the danger of identity-misbinding (i.e., requesting a private identity key for a identity that does not belong to the requestor), each service URL possesses the account name of its owner as an infix. For an account name of `JohnDoe`, a valid service URL is of the form `.../JohnDoe/...`. If a service requests a private key for a URL with infix `JohnDoe`, it has to prove (via an IBC signature) that it is in valid possession of the specified account. Figure 3 illustrates the activity chain for identity key retrieval.

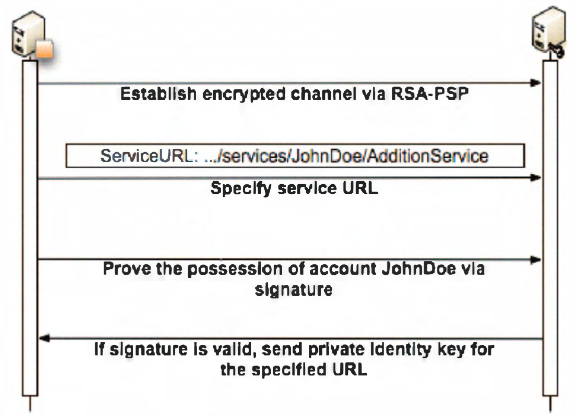
Establish encrypted channel via RSA-PSP
ServiceURL: .../services/JohnDoe/AdditionService
Specify service URL
Prove the possession of account JohnDoe via signature
If signature is valid, send private identity key for the specified URL

Fig. 3. A secure way to retrieve a service's private key for its service URL

To enable the service to issue such a signature, the service needs the private key for the identity `JohnDoe`. Since relaying the actual private identity key of the user to each service would be a critical security damage, a form of delegation is needed.

Delegation and Key Lifetime with Certificates: When using certificates, delegation is a well-known feature used by GSI. Instead of passing the original certificate to the delegate, so-called *proxy certificates* are utilized. To issue a proxy certificate, the user generates a new certificate and a corresponding private key using a commonly known algorithm. Next, the user signs the certificate using this private key. Afterwards, the user hands over the pair to the service that needs to act on behalf the user. By specifying an expiration date in the certificate, the user can also easily determine the lifetime of the proxy certificate. As long as the signature on the proxy certificate is trusted by the remote party and the lifetime has not been expired, the service can use the proxy certificates to establish secure connections.

Delegation with IBC : Delegation in the case of IBC is equal to the need of being able to generate a signature on behalf of an identity ID without knowing the actual private key. First, the user generates a new set of public shared parameters. Afterwards, (s)he signs the set using his or her private identity key. Next, the user sends the signed parameters and the corresponding private key to the service that requires delegation. Algorithm 1 shows the corresponding execution. Line 1 is a call to $\mathsf{GeneratePSP(k)}$ to create public shared parameters according to the specification in Section IV-B.1 using a security parameter k that determines the bit length of the primes P and Q. These primes are also used to form the private key in line 2. After choosing a random integer r (Line 3), the user finally signs the created PSP (in Line 4).

Algorithm 1 CREATE DELEGATION CREDENTIALS
```
//ID is the identity string of the issuing user
//$d_{\mathsf{ID}}$ is the private identity key of issuing user
Input: $\mathsf{ID}, d_{\mathsf{ID}}$   Output: New PSP signed by ID: $\mathsf{PSP_{ID}}$
1. $\mathsf{PSP_{ID}} \leftarrow \mathsf{GeneratePSP(k)}$
2. $\mathrm{SP} \leftarrow \{P, Q\}$
3. $r \in_R \mathbb{Z}$
4. $\mathrm{Sig}(\mathsf{PSP_{ID}}, r, \mathsf{ID}) = \left(\mathrm{Sik}(\mathsf{ID}, H(\mathrm{PSP})r), \mathrm{PSP}, G^{Rr}\right)$
```

Whenever a service is forced to prove that it acts in a legitimate way on behalf of a user, it signs a current time stamp based on the PSP that were signed by the user. The associated action is shown in Algorithm 2. In Line 1, the service extracts an identity key for its service URL (optionally with key-lifetime, see below). After taking the current time stamp (Line 2) and a random integer (Line 3), the service finally signs the time stamp (Line 4).

Algorithm 2 CREATING A DELEGATION SIGNATURE
```
Input: $\mathsf{PSP_{ID}}$, SP   Output: $\mathrm{Sig}(t, r, \mathsf{ID_S})$
1. $d_{\mathsf{ID}_S} \leftarrow \mathsf{Extract}(\mathsf{ID}_S, \mathsf{PSP_{ID}})$
2. $t \leftarrow \mathsf{CurrentTimeStamp()}$
3. $r \in_R \mathbb{Z}_N$
4. $\mathrm{Sig}(t, r, \mathsf{ID_S}) = \left(\mathrm{Sik}(\mathsf{ID_S}, H(t)r), t, G^{Rr}\right)$
```

The output of Algorithm 2 is a time stamp signed by the service URL ID_S based on the public shared parameters that were issued and signed by the root user.

Key Lifetime with IBC : A public key in an IBC system can easily be extended to carry lifetime information, as shown by Boneh and Franklin [2]. If t is the time stamp indicating when the public key should become invalid, the identity is simply extended by concatenating t to the actual identity string. For example, a former service URL of `.../JohnDoe/AdditionService` adopts the form `.../JohnDoe/AdditionService:12/31/2010` when being valid until 31th of December, 2010. It should be considered that each participant who wants to communicate with the service `.../JohnDoe/AdditionService` must know the validation time, since the extended identity is used in the computation of the session key. Only if the correct time stamp is concatenated, the key agreement will succeed. However, this is not a real obstacle; submitting the time stamp t in plaintext in combination with the Sik during the agreement is without risk and poses no security flaw.

Whenever the service needs a proxy-key, it uses its delegation signature capabilities (as detailed above) to request a new identity with a given validation time.

V. Security Overhead

IBC does not only reduce the complexity in a dynamic environment, but also reduces bandwidth. Details on the bandwidth savings are given below.

Certificate Approach. In a certificate approach, the following information must be transmitted: (1) The certificate including the user's public key, the signature of the CA and the additional parameters of the X.509 certificate. (2) A signed piece of data that can be verified by a third party using the user's private key to prove legitimate possession of the public key. Thus, we have 2048 bits (user's public key) + 2048 bits (signature of CA) + 5000 bits (additional X.509 parameters) + 2048 bit (proof signature) = 11144 bit in total. The above bit length can be reduced to 6144 bits by omitting the additional X.509 parameters which could partially be hard-coded into the verification system.

Identity-based Cryptography. In our approach, the binding between key and service address is done implicitly by the mathematics creating the signature, and thus no certificate is needed. The total bit length is: 2048 bits $\mathsf{Sik}(\mathsf{ID}, H(m)r)$, 32 bit ($T$), 2048 bits ($G^{Rr}$) = 4128 bits. This results in a reduction of factor $\sim$ 2.7 compared to the standard certificate-based approach, and a factor of $\sim$ 1.5 compared to a certificate-based approach omitting the additional X.509 parameters mentioned above.

VI. Conclusions

In this paper, several problems of securing service-oriented applications in Cloud environments were discussed, and it was argued that GSI and the corresponding Web service standards are not suited for this task. A lightweight security infrastructure for service-oriented Cloud applications to overcome the problems of certificate-based solutions was presented. The approach allows complex service-oriented applications to be deployed and used on-demand in a secure manner without the need for complicated certificate management. The service URLs can be used as public keys, such that creating a secure connection to a service is easy. Our proposal makes use of a novel identity-based cryptographic system to avoid the management problems of certificate-based security infrastructures. It also offers a completely independent setup of security domains and does not require a trust hierarchy compared to other IBC systems. Furthermore, a comparison with respect to data transfer between traditional approaches and identity-based cryptography was presented. The solution is both more flexible and easier to use than certificate-based security. It is also more efficient with respect to the network load.

VII. Acknowledgements

This work is partly supported by the German Ministry of Education and Research (BMBF) (D-Grid Initiative).

References

[1] Amazon Web Services LLC, Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2/.

[2] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computation*, 32(3):586–615, 2003.

[3] L. Chen, H. W. Lim, and W. Mao. User-Friendly Grid Security Architecture and Protocols. In *13th International Workshop on Security Protocols Workshop, Cambridge, UK*, pages 139–156, 2005.

[4] J. Crampton, H. W. Lim, and K. G. Paterson. What can Identity-Based Cryptography offer to Web Services? In *SWS '07: Proc. of the 2007 ACM Workshop on Secure Web Services*, pages 26–36. ACM, 2007.

[5] T. Dörnemann, M. Smith, and B. Freisleben. Composition and Execution of Secure Workflows in WSRF-Grids. In *8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 08)*, pages 122–129. IEEE Press, 2008.

[6] R. Gennaro, H. Krawczyk, and T. Rabin. Okamoto-Tanaka Revisited: Fully Authenticated Diffie-Hellman with Minimal Overhead. In *In preparation, IACR Preprint Server,*, 2010. http://eprint.iacr.org/2010/068.

[7] H. Li, Y. Dai, L. Tian, and H. Yang. Identity-Based Authentication for Cloud Computing. In *CloudCom '09: Proceedings of the 1st International Conference on Cloud Computing*, pages 157–166. Springer-Verlag, 2009, Beijing, China.

[8] H. W. Lim and K. G. Paterson. Identity-Based Cryptography for Grid Security. In *E-SCIENCE '05: Proc. of the First International Conference on e-Science and Grid Computing*, pages 395–404. IEEE Press, 2005.

[9] H. W. Lim and M. J. B. Robshaw. A Dynamic Key Infrastructure for Grid. In *EGC 2005 – European Grid Conference on Advances in Grid Computing*, pages 255–264, 2005.

[10] C. Schridde, M. Smith, and B. Freisleben. An Identity-Based Key Agreement Protocol for the Network Layer. In *SCN – The 6th Conference on Security and Cryptography for Networks*, volume 5229 of *Lecture Notes in Computer Science*, pages 409–422. Springer Verlag, 2008, Amalfi, Italy.

[11] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO 1984 Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Verlag, 1984.

[12] M. Smith, T. Friese, M. Engel, B. Freisleben, G. Koenig, and W. Yurcik. Security Issues in On-Demand Grid and Cluster Computing. In *Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRID 06)*, pages 14–24. IEEE Press, 2006.

[13] M. Smith, T. Friese, and B. Freisleben. Intra-Engine Service Security for Grids Based on WSRF. In *Proc. of the 2005 IEEE/ACM International Sysmposium on Cluster Computing and Grid (CCGRID'05), Cardiff, UK*, pages 644–653. IEEE Press, 2005.

[14] M. Smith, M. Schmidt, N. Fallenbeck, T. Dörnemann, C. Schridde, and B. Freisleben. Secure On-Demand Grid Computing. In *Journal of Future Generation Computer Systems*, pages 315–325. Elsevier, 2008.