ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Certificateless Identity-Based Proxy Signature for Grid Computing Authorization

Mohamed Amin JABRI and Satoshi MATSUOKA

May 2010, C–272

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Abstract

In this paper we propose a new Identity-based certificateless proxy signature transferable to a chain of proxy signers enabling fine-grained delegation, Authentication and Authorization decoupling, and Attributes-based Authorization within the Grid environment without relying on any kind of PKI certificates, proxy certificates or attribute certificates. Security and performance analysis of the proposed scheme are also discussed.

1 Introduction

Recently, Grids gained a great interest and a wide adoption among the academia and the industrial environment, thereby, resulting in an unprecedented growth in the number of grid computing users. In order to be able to use the grid, users must possess a valid grid credential. Generally, acquiring such a credential is time consuming, and sometimes requires future users to go through an in-person interview, which can be discouraging in some areas without a Grid Certificate Authority's representative or Certificate Registration authority. Also, while the number of Grid users is exploding, Grid administrators are facing a real concern in securely distributing, managing users credentials and revoking the compromised ones. Subsequently, Public Key Infrastructure (PKI) certificates and proxy certificates, which are the core components of the *de facto* standard GSI[1] security framework — in providing: mutual authentication, Single-Sign-On and delegation — are becoming an impediment to the Grid's scalability.

These shortcomings have led many researchers to propose new certificate-free security frameworks that still provide the required mutual authentication, Single-Sign-On and delegation. Some of these proposed schemes were based on the concept of Identity-Based Cryptography using pairings like in the works of [4, 13, 10, 20, 16, 14]. In this new paradigm, the problem of acquiring an authentic copy of a given user's public key to send him an encrypted traffic is no more part of the confidential traffic sender's responsibility. Instead, this problem is shifted to the traffic recipient side, thereby freeing the sender from any such burden. In fact, the public key of any given user will be any of his identifying strings such an e-mail address, or an IP address, etc. , which will remove the need to rely on PKI certificates in order to guarantee a genuine binding between a user identity and his randomly-generated public key. In addition, it will be the duty of the encrypted traffic recipient to acquire the corresponding decryption key or private key associated with the string used by the traffic sender as the recipient's public key. In identity-Based Cryptography systems, there will be a Trusted third-party Authority (TA) or Private Key Generator (PKG) which will be entitled to issue private keys for users after authenticating them and ensuring that their identities conform with the identifying string used as public key. This TA or PKG is the equivalent to the Certificate Authority (CA) in PKI which certifies the binding between a given user identity and his public key.

In conjunction with Identity-Based Cryptography, another key concept inline with the motivation to replace the standard delegation by means of proxy certificate is the signing capability delegation or proxy signature [15]. Indeed, being able to produce a genuine signature on behalf of an original signer without the need to share one's private key was the driving motivation behind the proxy signature concept. Generally, the original signer sends to his designated proxy signer a special signature, which can be used in turn by the intended proxy signer to compute a proxy signing key. It is to be noted that a modified verification equation rather than the one used to validate the original signer signature is used to check the authenticity of the proxy signer's signature. Moreover, any proxy signature's verifier convinced with the signature authenticity can assume that the signer is a designated and entrusted proxy signer by the original signer.

In this paper, we propose a novel authorization scheme [8] for the grid environment, through the use of Identity-Based cryptography and proxy signature concept. Although the integration of Identity-Based Cryptography (IDBC) within the grid security framework has already been proposed in literature, to the best of our knowledge we are the first to propose the use of the proxy signature concept within the

[1] GSI uses X.509 standard for certificates and proxy certificates

grid environment to enhance attribute-based authorization systems. Traditionally in such an attribute-based authorization system[2], users are required to acquire from a grid trusted Attribute Authority (like VOMS or CAS) a valid attribute certificate or attribute token, asserting in a certified manner their attributes, the roles they hold and their groups membership. Then, in order to access a given resource, the users push their attribute certificates (or their proxy certificates after embedding into it the attribute certificate) to the resource authorization manager which will make the authorization decision based on the presented attribute token. Still, our proposed scheme, lets users push their attributes to the resource authorization manager but without any use of attribute or proxy certificates. In fact, using a special variation of proxy signature, called warrant-based proxy signature scheme (kindly refer to section 2.3), we make the grid Attribute Authority, as an original signer, delegate his signing capability to any given authenticated user, as a designated proxy signer. This delegation will be performed under a tamper-proof warrant enclosing the user attributes and a validity time frame, which will be presented with any computed proxy signature as an authenticity requirement. Therefore, a resource authorization manager convinced by the warrant-based proxy signature authenticity can make authorization decisions based on the attributes carried within the presented signature's warrant. Furthermore, by doing so, we also achieve authorization and authentication decoupling. In fact, the authentication process will be carried out by the grid Attribute Authority which will enforce users' authentication before delegating his signing capability to them. Then, if we make the Attribute Authority include into the proxy signature warrant the authenticated user identity, the authorization process can be carried out by the resources' authorization manager, without further user authentication. This is due to the fact that validating a given warrant-based proxy signature that includes the involved parties identities (the proxy signer and the original signer identities) is equivalent to checking whether the proxy signer was approved by the original signer to be a valid proxy signer on his behalf, thereby implying the identity of the proxy signer being already authenticated by the original signer.

In addition, we enhanced the proxy delegation scheme, by allowing the proxy signing capability to be transferred and re-delegated to other designated proxy signers. This allows us to create a proxy signing delegation chain similar to the delegation chain with proxy certificates. To the best of our knowledge we are the first to propose a secure transferable proxy signature.

Our implementation of the proposed scheme, shows satisfying results (kindly refer to section 5.3) when considering a proxy delegation chain up to 40 nodes long, in terms of computational and bandwidth overhead.

The remainder of this paper is organized as follows. First, a review of the related work is given in section 2. Subsequently, in section 3, we explain the main motivations and merits of the proposed Certificateless Identity-Based Proxy signature scheme within the Grid environment. Then, in section 4, we introduce our proposed scheme, which include a description of the initial system setup, the private and public key establishment, and how proxy signature generation and verification are performed. After that, we provide, in section 5, an analysis of our proposed scheme. Lastly, section 6 concludes this work.

2 Related work

In this section, a review of the related work and building blocks of our proposed scheme is given. First, Identity-Based Cryptography including Hierarchical Identity-Based Cryptography is introduced. Then, Certificateless Public Key Cryptography including Hierarchical Certificateless public Key Cryptography is presented. Finally, Proxy Signing Delegation including certificateless warrant-based proxy signing is exposed.

2.1 Identity-based Cryptography

In Identity-Based Cryptography (IDBC) [14, 12, 3, 7] systems, any user's identifying string (like his e-mail or IP address) can be used as a public key. The corresponding private key is retrieved from a trusted

[2]we consider here only push-based authorization

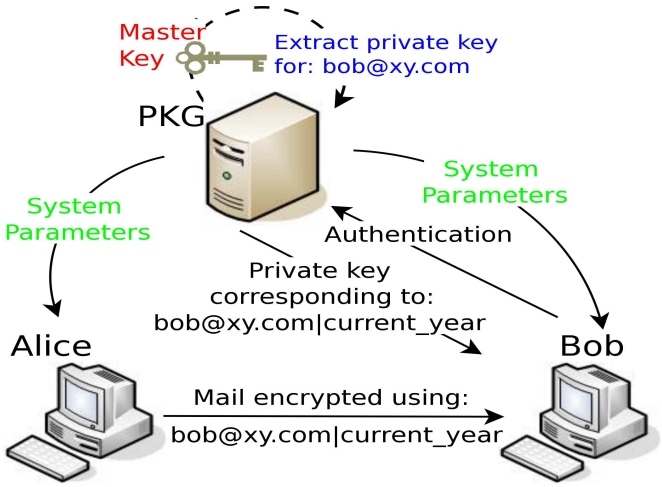


Figure 1: Identity Based Cryptography in mail confidentiality use

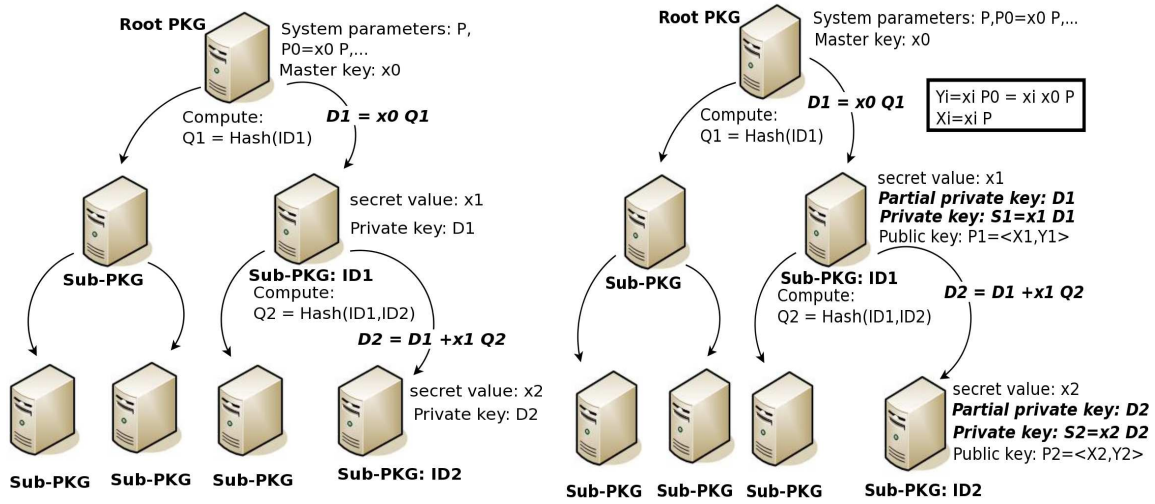
third party, called Private Key Generator (PKG). The PKG is responsible for extracting user's private keys through the use of a kept secret master key. It is to be noted that the extracted private keys should be delivered to their respective users over a secure and authenticated communication channel, to avoid keys disclosure. In addition, the PKG enforces user's authentication, prior to private key extractions. To illustrate the properties of IDBC, let us assume that a user *Alice* wants to send an encrypted mail to another user *Bob* at his e-mail address: *bob@xy.com*. As depicted in figure 1, *Alice* needs to know, in addition to *Bob*'s e-mail address, his PKG publicly published system parameters, in order to encrypt the e-mail. When *Bob* receives the encrypted mail, he contacts his PKG and requests the private key corresponding to his public key (*bob@xy.com—current_year* in this example). After authenticating *Bob*, The PKG uses his master key to extract *Bob*'s private key and sends it back in a secure way to *Bob*, who can then read his e-mail. In this example, we can see that the IDBC scheme alleviates *Alice* from the burden of getting an authentic copy of *Bob*'s public key as being required in standard Public Key Infrastructure based systems. Therefore IDBC transforms the problem of public key distribution (in PKI) to the secure distribution of private keys (key extraction by the PKG) corresponding to any user identifying string like the user's e-mail address.... In addition, key expiration in IDBC is straightforward. As illustrated in the previous example (see figure 1), *Bob* can use his private key (corresponding to *bob@xy.com—current_year*) only during the current year. Therefore, Once a year *Bob* needs to acquire a new private key from the PKG. More granularity can be set by adding *current_year—current_month* to Bob's e-mail address as his public key. Thus, resulting in a monthly refreshed key system. The main advantage with IDBC key revocation [3] is that *Alice* is no more required to obtain a new certificate from *Bob* every time he refreshes his keys.

The concept of IDBC was first introduced, in 1984, by Adi Shamir to simplify certificate management in e-mail systems. But, the first fully functional and practical IDBC system is due to the efforts of Boneh and Franklin [3] who successfully proposed an IBE using bilinear map[4] (for a mathematical definition of bilinear pairing, please refer to section 4) between two groups. In Boneh and Franklin's proposed scheme, all users are required to contact and authenticate with a single and unique PKG in order to have their private keys issued. This can lead to performance degradation, especially in large scale deployment scenario like in the Grid. In fact, If we consider that extracting private keys is computationally expensive [6] and that PKG is responsible for verifying users' identities before delivering the private keys (over secure channels with the intended users), it is obvious that the PKG will become a bottleneck. In order to address this performance shortcoming, Gentry and Silverberg came up with a fully Hierarchical

[3] here key revocation through key expiration

[4] they used the Weil pairing on elliptic curve as example of such a map

Identity-Based Encryption and Signature scheme (HIBE and HIBS) [6]. In their proposed HIBE and



(a) Gentry-Silverberg HIBE/HIBS

(b) Hierarchical CL-PKC. ($Y_i = x_0 X_i = x_0 x_i P$)

Figure 2: Key establishment in Gentry-Silverberg HIBE/HIBS and in Hierarchical CL-PKC.

HIBS scheme, a root or system PKG distributes the workload by delegating private key generation and identity authentication to lower-level PKGs or sub-PKGs, which are organized in a tree-like hierarchy (see figure 2a). At each level of the tree, a sub-PKG uses its acquired private key, as a master key (like in Boneh and Franklin scheme), and extracts private keys for their child sub-PKGs. For instance, as depicted in figure 2a, the sub-PKG with the identity ID_1, upon receiving his private key D_1 from his parent (the root-PKG), extracts the private key D_2 for his child: the sub-PKG having ID_2 as an identity. It is to be noted that in HIBE and HIBS there is only a unique set of system parameters which are generated and publicly published by the root PKG (like in the Boneh and Franklin scheme). All the sub-PKGs below the root level do not generate any set of system parameters. Instead, they use the set of system parameters generated by the root PKG. Also, as one can see in the example of figure 2a, each parent have access to the private keys of the sub-PKGs situated below them. This property, is called *key escrow* and is inherent to identity-based cryptosystems.

2.2 Certificateless Public Key cryptography

The Certificateless Public Key Cryptography (CL-PKC) [1] is a model for using public key cryptography without requiring the use of certificates (as in PKI) and yet do not suffer from the built-in *key escrow* shortcoming of Identity Based Cryptography. The CL-PKC scheme owes much to the Boneh and Franklin proposed scheme [3] but with the introduction of some modifications which obviates the hindering *key escrow* property. In fact, CL-PKC still make use of a trusted third party, for key generation like the PKG in Boneh and Franklin scheme, with the exception that the CL-PKC 's PKG has no more access to entities private keys. For instance, let us consider a user Bob with an identity ID_B (like in the previous section's example ID_B is a well chosen identifying string for Bob), to illustrate public/private keys generation within the CL-PKC scheme. After a successful authentication procedure with the PKG, Bob is supplied with a *partial private key* (instead of a private key like in the original Boneh and franklin scheme) D_B which the PKG computes from the identifier ID_B and a master Key. Then, Bob combines his partial private key D_B with some secret information x_B to generate his actual private key S_B. Therefore, Bob's private key S_B is no more available to the PKG, which eliminates the *key escrow* from the system. It is to be noted, that similarly to Boneh and Franklin scheme, partial private keys have to be delivered to their intended users over an authenticated and confidential communication channel.

Also, Bob combines his secret information x_B (the same used for S_B generation) with the system's parameters (publicly published by the PKG) to generate his *public key* P_B. This latter can be made available to other users by transmitting it along with messages, especially in a signing application, or by placing it in a public directory [1]. Unlike PKI, no further security mechanisms like certificates are applied to protect public keys in CL-PKC. For example, in order to encrypt a message to Bob or verify a signature from Bob, other users makes use of both Bob's public key P_B and Bob's identifying string ID_B for encryption and signatures verification. A formal and full description of the CL-PKC encryption and signing procedures can be found in [1].

Similarly, as stated in the previous section, when considering a large scale deployment, like in the grid environment, the PKG most likely will become a bottleneck. Thus, the authors of [1] proposed a hierarchical CL-PKC (HCL-PKC) similar to the Gentry and Silverberg HIBE/HIBS. In HCL-PKC the system PKG or root PKG distributes the load by delegating the partial private keys extraction and identity authentication to lower-level PKGs or sub-PKGs. Like in HIBS/HIBE, sub-PKGs are organized in a tree-like fashion with the system's PKG as its root. The sub-figure 2b depicts an example showing how HCL-PKC's keying materials are established. By comparing the sub-figures of figure 2, we can notice the main differences in keying materials distribution between HIBE/HIBS and HCL-PKC and how *key escrow* was removed in this latter.

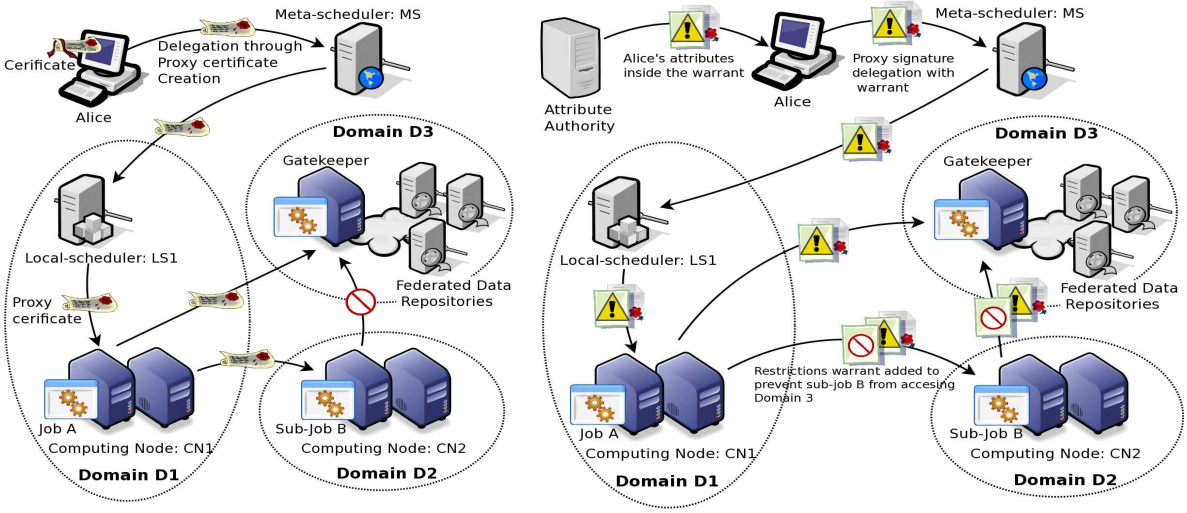
2.3 Proxy signing delegation

The concept of Proxy Signature was first introduced, in 1996, by Mambo, Usuda and Okamoto [15]. The Proxy Signature scheme allows an entity, called *original signer*, to delegate another entity, called a *proxy signer*, the ability to sign messages on its behalf. According to the authorization degree, Proxy Signatures are classified into three types: full delegation, partial delegation and delegation by warrants [15]. For example, let us assume that Olga, an original signer, asks Peggy to be her proxy signer and carry out signing instead of her, and Veronica being a third party trying to verify the authenticity of the produced signatures.

In full delegation, the proxy signer, Peggy, is given the same signing secret key as the original signer, Olga. Thus, Peggy can deliberately create the same signature as Olga over any message. This property leads to the trouble that both Peggy and Olga can repudiate their signature, claiming that the other signer has created it [15]. In the partial delegation type, a new signing secret, derived from Olga's signing key, is given in a secure way to Peggy. The verification of a signature created by Peggy follows a modified verification equation rather than the one used for Olga's signature verification. Thus, signatures created by both Peggy and Olga over a given message are totally different and distinguishable [15]. In this type of signing's delegation, the proxy signer, Peggy, after receiving the proxy signing key, will able to sign any message on behalf of the original signer, Olga, without any restrictions like the approved scope of messages to be signed or the time period to output a genuine signature. The last delegation type, is based on the use of a warrant. This latter, for instance, certifies that Peggy is a signer to be entrusted and that she is the designated proxy signer to sign messages on behalf of the original signer, Olga. The signing delegation by warrant can be implemented using two approaches. In the first approach, the warrant is composed of a message part stating the identity of Peggy and the Olga's signature over the public key of Peggy. In the second approach, the warrant is composed of message part containing Peggy's identity and Olga's signature over a newly generated public key. The corresponding private key is securely transmitted to Peggy. In both approaches, the proxy signer Peggy, use the private key in conjunction with the warrant to produce the proxy signature over a given message.

The authors of [11] proposed a certificateless warrant-based proxy signature scheme, which makes use of CL-PKC for keys distribution and the second approach of signature delegation by warrant. For example, after computing public and private keys using CL-PKC scheme, Olga creates a warrant stating her identity as being the original signer and the identity of Peggy as the designated trusted proxy signer along with the delegation period and any delegation restrictions. Then, Olga signs the delegation warrant with her private key[5] and sends the signature in a secure way to Peggy. By combining her private key

[5]here it is the CL-PKC private key



(a) Using Certificate and Proxy certificate approach

(b) Certificateless approach (using our proposed scheme)

Figure 3: A Grid environment use-case

with Olga's signature over the delegation warrant, Peggy generates the proxy signing key which will be used to sign messages on behalf of Olga, the original signer. Each time Peggy signs a message on behalf of Olga, the delegation warrant needs to be added to the proxy signature. In fact, in order to check the authenticity of Peggy's proxy signature over a given message, Veronica starts, first, by enforcing the restrictions stated in the warrant. Then, she uses the public keys of both Olga and Peggy, the warrant and the system public parameters to verify the authenticity of the proxy signature. Interested readers, may refer to [11] for the formal description of the proxy signature generation and verification within the warrant-based proxy signature scheme.

3 Motivations

In this section we will depict the main motivations and advantages behind the adoption of our proposed scheme (see section 4) compared to the existing standard approaches. In fact, The first and main driving motivation behind the proposed scheme was to avoid using PKI certificates within the grid environment in order to alleviate the burden of distributing, managing and using PKI certificates and proxy certificates. The second intended goal from our scheme, is to offer a flexible, transferable (if permitted), fine-grained delegation. Also, as far as delegation chain's validation, we intend to over perform than the traditional grid X.509 proxy chain creation and validation (see section 5). Finally, our proposed scheme decouples authentication and authorization. This is done by moving the authentication phase from the resource level to the VO or campus authentication server –using password-based, certificate-based or any other, already, in use authentication's approach. Thus, letting resources to enforce authorization based on user's attributes within the VO.

3.1 Certificateless approach

The driving motivation behind our work, was to mitigate the burden of using and maintaining public key certificates, while still providing Single-Sign-On (SSO) and delegation. Proxy certificates are the *de facto* standard in achieving SSO and delegation within the Grid environment. For example, as illustrated in sub-figure 3a, the user Alice delegates her capabilities to the Meta-scheduler MS, when submitting her job to be executed on the grid, through the creation of a proxy certificate. In fact, the

MS, after authenticating and receiving Alice's job, generates a new pair of public/private key and sends a Certificate Signing Request (CSR) containing the newly generated public key to Alice. Using her certificate's private key, Alice signs the CSR and send it back to the MS. At this point, the MS is in possession of a valid proxy certificate signed by Alice and then can authenticate and submit Alice's job on her behalf for execution on available grid resources or even re-delegate this task to a local-scheduler. It is to be noted that, at any time, no more intervention from Alice is required once she signed the proxy certificate for MS.

Similarly, In our proposed scheme we intend to provide SSO and delegation without using any kind of certificate. As depicted in sub-figure 3b, the same user Alice submits her job to MS like in the previous example of sub-figure 3a, but instead of any proxy certificate creation, Alice delegate her signing capability to MS using a certificateless warrant-based proxy signature (refer to section 2.3). At this point, the MS can act on Alice behalf by signing any message or job request using the proxy signature scheme. MS can further re-delegate Alice capabilities to the local-scheduler $LS1$ as shown if sub-figure 3b, by transferring the signing capability on Alice behalf to $LS1$. To the best of our knowledge, we are not aware of any work on the literature that addressed a *multi-level proxy signature delegation*, in which a proxy signer can further delegate the acquired proxy signing capability from an original signer to another proxy signer and so forth.

3.2 Fine-grained delegation

In the existing production grids, delegation is performed through the use of proxy certificates. For example when an Entity B needs to act on behalf of another entity A, B generates a new proxy certificates having one of the predefined delegation types, and sends it to A to be signed. If A approves the requested delegation, then A signs with his long term private key the new proxy certificates and sends it back to B. It is obvious, to note that such kind of delegation is not flexible and is very restrictive. In figure 3a, for example, job A running on computing node CN_1 cannot prevent the sub-job B at domain D_2 to authenticate with the gatekeeper at Domain D_3 and access the data repositories.

With our proposed scheme we are owing to target fine-grained and restricted enabled delegation. For instance as shown in the figure 3b, job A running on the Computing Node CN_1 within Alice's delegation chain can restrict and fine-tune delegation to sub-job B that will be running on the computing node CN_2. For example, let us consider that job A wants to prevent all the sub-jobs it instantiates in domain D_2 to access the data repositories on domain D_3. To do so, job A can include the intended delegation restrictions and the applicable set of rules or policies to be enforced by the Gatekeeper at domain D_3, within the warrant amending the proxy signature delegation to sub-job B. If this latter tries to access any resources within domain D_3, he will be blocked by its gatekeeper. In fact, in our scheme producing a proxy signature (in order to authenticate a resource allocation or access) is subject to the content of all the warrants of the delegation chain entities and their carried restrictions. It is to be noted that it is the responsibility of the proxy signature verification entity to enforce all warrants specified restrictions and policies.

3.3 Enhanced delegation's chain creation and validation

Within the GSI framework it is common that delegation (creation of new proxy certificate) can be chained [17]. This is due to the fact that, within a grid systems, allocation and access to a resource can be made either, directly, by a user or, indirectly, by a process acting on a user's behalf [5]. For instance, as depicted in figure 3a, the user Alice can delegate her privileges to the meta-scheduler MS, to allow her submitted job to be executed and scheduled on her behalf. Then, the meta-scheduler MS re-delegate Alice acquired privileges to the local scheduler LS_1 located at Domain D_1. LS_1, in turn, lunches Alice job on the computing node CN_1 and grant it all Alice capabilities for the data repositories access at domain D_3. Thus, resulting in a chain of a proxy certificates, with respect to the GSI framework. In order to validate such a proxy delegation chain, The gatekeeper have to check the validity and authenticity of each proxy certificates up to the public key certificate of the initiating user

Alice, which can be computationally intensive (due to the use of RSA), especially when the delegation chain is very long. Also, it is to be noted that the creation of a new proxy certificate, for delegation purpose, is extremely prohibitive. This is due to the computationally demanding process of generating fresh RSA public/private keys.

In our proposed scheme, as depicted in figure 3b, the local scheduler LS_1 can delegate its acquired privilege from Alice through the meta-scheduler MS to the job A that will be running on compute node CN_1, by just issuing a delegation signature. Job A, then, can use the delegation signature of LS_1 as an input to either, further generate a new delegation signature to another sub-job B, or to compute the proxy signing key (refer to section 4) used to sign messages on behalf of LS_1. Which enables us to create the same delegation chain as in the GSI framework. On the other hand, because we are relying on a form of certificateless public key cryptography in generating signatures and computing keys, the setup and validation of a delegation chain will be computationally efficient compared to the way it is done within the GSI framework.

3.4 Authorization and authentication decoupling

In the GSI framework, authorization and authentication are both carried at the resource level. In fact, at first the resource authenticates the access requesting entity by validating and checking the requesting entity's public key certificate or proxy chain certificates. Then an access authorization decision is made based on the requesting entity distinguished Name (DN) or the embedded attribute certificates within a proxy certificate.

In our proposed scheme we intend to move the authentication from the resource level, using a fixed predefined authentication procedure, to the VO Attribute Authority (AA) level. In fact, the authentication will be carried by the AA, using any suitable authentication procedure (certificate-based, password-based ...). If successfully authenticated, the AA will delegate his signing capabilities to the corresponding user by issuing him a warrant-based delegation signature (refer to section 4). Hence, when a resource validates a proxy signature (in our scheme) over a job request, it can be sure that the job requesting entity is a valid designated proxy signer within a delegation chain acting on behalf of an, already, authenticated user by the VO's Attribute Authority. Furthermore, if we make the AA to enclose the VO attributes of authenticated users within the delegation signature warrant, then the resource after validating the proxy signature can extract and make authorization decision based on those attributes (all warrants are parts of generated signatures in our scheme). Also, The carried attributes authenticity is guaranteed because our proposed scheme takes into consideration all warrants digest (or hash) when generating delegations or proxy signatures.

4 Proposed Scheme

The building blocks of our proposed Scheme are based on Gentry-Silverberg Hierarchical ID-Based Cryptography [6], Certificateless Public Key Cryptography [1], and Proxy signature [11] from Bilinear pairings.

In the remainder of this section, we describe our proposed scheme which consists of five basic procedures or steps. The first two ones, *System initial Setup* and *Root and lower level setup* are responsible for initialization, public and private keys establishment. In the *Signing capability delegation* step, delegation of the signing capability of an original signer to a designated proxy signer and how such a delegation is further transferred to other other proxies is explained. Finally, the *Proxy signing* and *Proxy Signature verification* procedures demonstrate the way to proxy-sign a given message and how its authenticity can be checked by a verifier. We, also, give a brief definition to bilinear pairings, which is the foundation to pairing based cryptography and hence to our proposed scheme. Figure 4 illustrates the adopted architecture and entities involved in our proposed scheme.

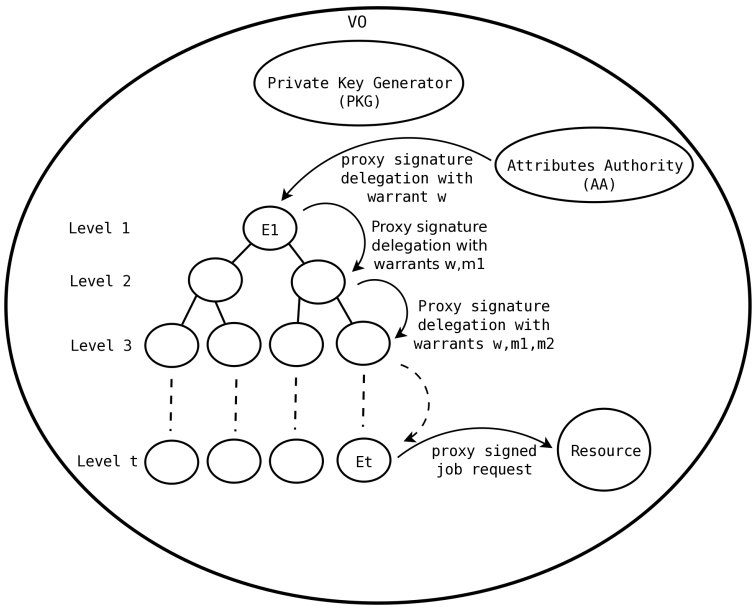


Figure 4: Proposed architectural scheme

4.1 Bilinear pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups having the same order q. In the following, $\mathbb{G}_1$ will be denoted additively whereas $\mathbb{G}_2$ will be denoted multiplicatively.

Definition 1 (admissible bilinear pairing)**.** *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be 2 groups as mentioned above. An admissible bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ which verifies the followings three properties:*

- *Bilinear: given $P, Q, R \in \mathbb{G}_1$ we have $e(P, Q + R) = e(P, Q) \cdot e(P, R)$ and $e(P + Q, R) = e(P, R) \cdot e(Q, R)$. Hence, for all $a, b \in \mathbb{Z}_q$, $e(aP, bQ) = e(P, Q)^{ab}$*
- *Non degenerate: there exists $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1_{\mathbb{G}_2}$*
- *Computable: given $P, Q \in \mathbb{G}_1$ we can efficiently compute $e(P, Q) \in \mathbb{G}_2$.*

In practice, the bilinear map e can be implemented using the Weil [3] or the Tate [2] pairings on elliptic curves over a finite field.

4.2 Security assumptions

The security of our proposed scheme relies on some security assumptions stipulating the hardness in resolving some computational problems. These latter are the *Discrete Logarithm problem* (DLP), the *Bilinear Diffie-Hellman problem* (BDHP), and the *Computational Diffie-Hellman problem* (CDHP) [4]. In the following, we give a brief definition of these computational problems.

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an *admissible bilinear pairing*, with $\mathbb{G}_1$ and $\mathbb{G}_2$ being two cyclic groups of a prime order q (q is k bits long).

DLP: For a given $P, aP \in \mathbb{G}_1$ find the unknown value of $a \in \mathbb{Z}_q$

BDHP: For a given $P, aP, bP, cP \in \mathbb{G}_1$ with $x, y, z \in \mathbb{Z}_q$ being unknown, compute the value of $e(P, P)^{xyz} \in \mathbb{G}_2$

CDHP: For a given $P, aP, bP \in \mathbb{G}_1$ with $a, b \in \mathbb{Z}_q$ being unknown, compute the value of $abP \in \mathbb{G}_1$.

We assume in our scheme that these problems are intractable in time in polynomial in k.

4.3 System initial setup

The trusted authority or private key generator (PKG), given a security parameter k, chooses two groups of a prime order q (q is k bits long): $\mathbb{G}_1$ being an additive cyclic group and $\mathbb{G}_2$ being a multiplicative cyclic group. In addition, the PKG chooses an admissible bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, and three hash functions: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q$ and $H_3 : \{0,1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q$. It is to be noted that H_1, H_2 and H_3 have to be a one way and a collision-resistant hash fuctions, as a system security requirement. Subsequently, the PKG picks up a random $x_0 \in \mathbb{Z}_q$, which will be kept secret as a *master key*, and a random generator $P = X_0 \in \mathbb{G}_1$. Then, the PKG computes $P_{pub} = x_0 P = Y_0$, and publicly publishes the system parameters: $param = (G_1, G_2, e, H_1, H_2, H_3, q, P_0)$, with $P_0 = \langle X_0, Y_0 \rangle$ as the PKG's public key.

4.4 Root and lower level setup

Each entity E_i at a level i (including the Attribute authority AA at level 1) having the identity tuple $(ID1||\ldots||ID_i)$ (ID_A for the AA server) chooses a random value $x_i \in \mathbb{Z}_q$ (x_A for the AA) and keeps it as a secret value.

4.4.1 Partial private key extraction

The ancestor E_{t-1} (including the PKG) of each entity E_t at the level $t \geq 1$, extracts the partial private key D_t using his partial private key D_{t-1} (previously acquired from his corresponding ancestor) and his secret value x_{t-1} as follows:

$$\begin{aligned} Q_t &= H_1(ID_1 \parallel ID_2 \parallel \ldots \parallel ID_t) \\ D_t &= D_{t-1} + x_{t-1} Q_t = \sum_{i=1}^{t} x_{i-1} Q_i \end{aligned}$$

Then, E_{t-1} sends to each entity E_t its partial private key D_t over a secure and authenticated channel.

4.4.2 Private and public key establishment

After receiving his partial private key D_t each node E_t computes his private key $S_t = x_t D_t$, and public key $P_t = \langle X_t, Y_t \rangle$, with $X_t = x_t P$ and $Y_t = x_t P_{pub} = x_t x_0 P = x_0 X_t$. Thereby, we will have:

$$S_t = x_t D_t = \sum_{i=1}^{t} x_t x_{i-1} Q_i \tag{1}$$

To ensure that the received value of D_t has been correctly computed, E_t can check whether the following equation[6] holds:

$$e(D_t, P) = \prod_{i=1}^{t} e(Q_i, X_{i-1})$$

Otherwise, E_t asks his ancestor to extract once again the partial private key D_t and resend it back.

4.5 Signing capability delegation

First, the Attribute Authority (AA) starts by delegating his signing capability to the entity E_1 at the level 1. AA generates a delegation warrant, signs it with his private key and sends the signature along with his public key to E_1. Subsequently, E_1 can redelgate the acquired signing capability to another

[6] The set of $(P_i = \langle X_i, Y_i \rangle)_{0 \leq i \leq t-1}$ can be obtained from E_t's parent when receiving his partial private key D_t

entity E_2 at a different level by generating a new delegation warrant which E_1 signs using both his private key and the previously received delegation signature from AA. Upon receiving and validating the delegation signature, E_2 can further redelegate the acquired signing capability on behalf of AA and E_1 to another entity E_3 and so on. In our scheme we distinguish between the first signing capability delegation from AA to E_1 and its further redelegation to other entities $(E_i)_{i\geq 2}$. In the following, We describe these two kinds of signing capability delegation.

4.5.1 Attribute Authority signing delegation

The attribute Authority server (AA), after authenticating the user E_1, extracts his VO's attributes from the users attributes database and generates the warrant m_A. This warrant includes the identities of both, the AA server and E_1, an expiry date after which the warrant m_A become obsolete in addition to E_1 attributes which will be used for authorization[7] decision making at the resource level. In addition, AA includes a copy of his public key P_A within the warrant. Then, the server picks-up a random $r_A \in \mathbb{Z}_q$, and outputs the delegation signature $\sigma_A = \langle m_A, U_A, P_A, V_A \rangle$ as follows:

$$\begin{aligned} U_A &= r_A Q_A \\ h_A &= H_2(m_A, U_A) \\ V_A &= (h_A + r_A) S_A \end{aligned}$$

and sends σ_A to E_1 over a secure channel[8].

To ensure that σ_A was correctly computed by the AA, E_1 can check whether the following equations hold:

$$\begin{aligned} e(Y_A, P) &= e(X_A, P_{pub}) \\ e(V_A, P) &= e(h_A Q_A + U_A, Y_A) \end{aligned}$$

Otherwise, E_1 needs to reauthenticate with the AA and get a new delegation signature σ_A.

4.5.2 Proxy signing redelegation

After receiving and validating σ_A, E_1 generates a delegation warrant m_1 in which he includes his identity as well as E_2 identity (as his proxy) and the delegation restrictions he wants to be enforced. Also, he includes a copy of his public key P_1 within m_1. Then, he picks up a random $r_1 \in \mathbb{Z}_q$ and computes:

$$\begin{aligned} W_1 &= r_1 x_1 Q_1 = w_1^1 \\ h_1 &= H_2(m_1, Q_1) \\ V_1 &= r_1 h_1 S_1 + h_1 V_A \end{aligned}$$

and sends $\sigma_1 = \langle m_A, U_A, P_A, m_1, W_1, P_1, v_1 \rangle$ to E_2 over a secure channel to E_2.

In general, each node E_i after receiving σ_{i-1} from his ancestor E_{i-1}, generates a delegation warrant m_i containing his identity as well as the identity of his designated proxy signer E_{i+1} and the delegation restrictions to be enforced over his delegation in addition to his public key P_i. Then, he chooses a random $r_i \in \mathbb{Z}_q$ and computes:

$$\begin{aligned} W_i &= (r_i x_i Q_k)_{1\leq k\leq i} = (w_k^i)_{1\leq k\leq i} \\ h_i &= H_2(m_i, Q_i) \\ V_i &= r_i h_i S_i + h_i V_{i-1} \end{aligned} \tag{2}$$

Subsequently, he sends the computed signature $\sigma_i = \langle m_A, U_A, P_A, (m_j)_{1\leq j\leq i}, (W_j)_{1\leq j\leq i}, (P_j)_{1\leq j\leq i}, V_i \rangle$ to E_{i+1}, over a secure channel.

[7] attribute push based authorization

[8] the secure channel use is only required to protect the content of m_A from unwanted discloser, thereby guaranteeing the privacy of E_1. Otherwise, a public channel can be used.

Thus, after $t+1$ delegations (from AA to E_{t+1}), we can infer that V_t, within the delegation signature σ_t, satisfies the following equation:

$$\begin{aligned} V_t &= \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} h_j \right) r_i h_i S_i + \left(\prod_{j=1}^{t} h_j \right) V_A \\ &= \sum_{i=1}^{t} \left(\prod_{j=i}^{t} h_j \right) r_i S_i + \left(\prod_{j=1}^{t} h_j \right) (h_A + r_A) S_A \end{aligned} \tag{3}$$

It is to be noted that each entity $(E_t)_{t\geq 2}$ can validate a received delegation signature σ_{t-1} as follows. First, E_t extracts the identities tuples within the warrants $(m_i)_{1\leq i\leq t-1}$ and m_A, and checks if the original signer or delegator stated within a warrant m_i is also stated as a proxy signer within the previous warrant m_{i-1}. Thereby, forming the chain of identities tuples corresponding to the delegation chain. Using the so formed identity tuples chain, E_t computes $h_i = H_2(m_i, Qi)$ for all $1 \leq i \leq t-1$. Finally, the delegation signature will be considered as valid[9], if the following equations[10] hold:

$$\begin{aligned} e(Y_A, P) &= e(X_A, P_{pub}) \\ e(Y_i, P) &= e(X_i, P_{pub}) \text{ , for all } 1 \leq i \leq t-1 \\ e(V_{t-1}, P) &= \frac{\prod_{i=1}^{t-1} \prod_{k=1}^{i} e(w_k^i, X_{k-1})^{\prod_{j=i}^{t-1} h_j}}{e(h_A Q_A + U_A, Y_A)^{-\prod_{j=1}^{t-1} h_j}} \qquad (4) \\ &= \frac{\prod_{k=1}^{t-1} e\left(\sum_{i=k}^{t-1} (\prod_{j=i}^{t-1} h_j) w_k^i, X_{k-1} \right)}{e(h_A Q_A + U_A, Y_A)^{-\prod_{j=1}^{t-1} h_j}} \qquad (5) \end{aligned}$$

In this case, E_t will be also in possession of an authentic copy of the public keys $(P_j)_{1\leq j\leq t-1}$ of all the previous entities $(E_j)_{1\leq j\leq t-1}$.

4.6 Proxy Signing Scheme

To proxy sign a message m, first, E_t computes his proxy signing key $S_{Proxy_t} = V_{t-1} + r_t S_t$, with r_t being a random value in $\mathbb{Z}_q$ and V_{t-1} the value taken from the delegation signature σ_{t-1}. Subsequently, E_t chooses another random $a \in \mathbb{Z}_q$ and then computes:

$$\begin{aligned} W_t &= (r_t x_t Q_k)_{1\leq k\leq t} = (w_k^t)_{1\leq k\leq t} \\ R &= e(P, aP) \\ h_t &= H_2(m, Q_t) \\ h &= H_3(m, R) \\ S &= h h_t S_{Proxy_t} + aP \end{aligned}$$

Finally, E_t outputs the signature $\sigma_{Proxy_t} = \langle m_A, U_A, P_A, (m_i)_{1\leq i\leq t-1}, (W_i)_{1\leq i\leq t}, (P_i)_{1\leq i\leq t}, h, S \rangle$ over the message m.

4.7 Proxy Signature verification

Any verifier knowing the system parameters under which a given proxy signature $\sigma_{Proxy_t} = \langle m_A, U_A, P_A, (m_i)_{1\leq i\leq t-1}, (W_i)_{1\leq i\leq t}, (P_i)_{1\leq i\leq t-1}, h, S \rangle$ over a message m was computed, can check its authenticity

[9] the delegation restrictions, as delegation's validity time frame, within all the warrants must be enforced by E_t.

[10] $e(V_{t-1}, P)$ can be computed using either equation 4 or 5.

by computing:

$$\begin{aligned} h_j &= H_2(m_j, Q_j)\ ,\ \text{for all } 1 \leq j \leq t-1 \\ h_t &= H_2(m, Q_t) \\ h_A &= H_2(m_A, U_A) \\ r &= \frac{e(S,P) \cdot e(h_A Q_A + U_A, Y_A)^{-h \prod_{j=1}^{t} h_j}}{\prod_{i=1}^{t} \prod_{k=1}^{i} e(w_k^i, X_{k-1})^{h \prod_{j=i}^{t} h_j}} \end{aligned} \tag{6}$$

The verifier will accept the signature if all the delegation's restrictions expressed within the warrants set $(m_i)_{1 \leq i \leq t-1}$ and m_A are not violated, and if the following equations hold:

$$\begin{aligned} e(Y_A, P) &= e(X_A, P_{pub}) \\ e(Y_i, P) &= e(X_i, P_{pub})\ ,\ \text{for all } 1 \leq i \leq t \\ h &= H_3(m, r) \end{aligned}$$

5 Scheme analysis

In this section, we will prove the correctness of the proposed scheme, analyse its security and evaluate its performance with respect to computational and communication (signature size) costs.

5.1 Correctness

Claim 1 (scheme Correctness)**.** *Let $(P_i)_{1 \leq i \leq t}$ and P_A be a set of correctly computed public keys, and $\sigma_{Proxy_t} = \langle m_A, U_A, (m_i)_{1 \leq i \leq t-1}, (W_i)_{1 \leq i \leq t}, (P_i)_{1 \leq i \leq t}, h, S \rangle$ be a valid proxy signature over a given message m, which was produced following our proposed scheme. Then, the proxy signature verification equations:*

$$\begin{aligned} e(Y_A, P) &= e(X_A, P_{pub}) \\ e(Y_i, P) &= e(X_i, P_{pub})\ ,\ \textit{for all } 1 \leq i \leq t \\ r &= \frac{e(S,P) \cdot e(h_A Q_A + U_A, Y_A)^{-h \prod_{j=1}^{t} h_j}}{\prod_{i=1}^{t} \prod_{k=1}^{i} e(w_k^i, X_{k-1})^{h \prod_{j=i}^{t} h_j}} \end{aligned}$$

are correct and hold.

For the mathematical proof of the scheme correctness property, interested readers are kindly referred to appendix A.

5.2 Security concerns

The propose scheme does not suffer from the Identity-Based cryptography inherent *Key escrow* where the PKG, is in possession of users private keys. In fact, private keys are computed locally based on randomly generator secret value and the received partial private key (refer to section 4). Thus, our scheme is *key escrow free*.

According to [9, 19], a secure proxy signature should fulfill the following requirements: *strong unforgeability*, *verifiability*, *strong identifiability*, *strong undeniability* and *prevention of misuse*. In the following, the original signer is being the Attribute Authority and the proxy signer is being one of the $(E_i)_{i \geq 2}$ entities (refer to figure 4) and the proxy delegator is being the corresponding ancestor E_{i-1} of E_i.

Strong unforgeability: A legitimate proxy signer is the only entity to be able to create a valid proxy signature over a given message on behalf of an original signer. Even the original signer or the proxy delegator cannot forge such a proxy signature.

Verifiability: From a proxy signature, a verifier can be convinced of the agreement of the original signer and all the intermediate proxies within the delegation chain on the signed message.

Strong identifiability: From a proxy signature, a verifier can determine the identity of the corresponding proxy signer.

Strong undeniability: Once a proxy signer generates a proxy signature over a message on behalf of an original signer and all the intermediate proxy signers on the delegation chain, the proxy signer cannot repudiate his signature against anyone.

Prevention of misuse: We should be confident that the delegation signature and the proxy signing key are used only for delegating the signing capabilities and to proxy sign messages without violating any of the agreed upon delegation restrictions and warrants.

We will discuss the security of our proposed scheme according to these requirements.

5.2.1 Strong unforgeability

In our scheme, even a given ancestor entity E_i, who knows the partial private key D_{i+1} of the entity E_{i+1} and V_i in the proxy delegation signature, cannot compute neither a valid proxy signature $\sigma_{proxy_{i+1}}$ nor a valid delegation signature σ_{i+1} in place of E_{i+1}. To do so, E_i must acquire the secret value x_{i+1} or the private key S_{i+1} of E_{i+1} in order to compute V_{i+1} or the proxy signing key $S_{proxy_{i+1}}$. if E_i is able to learn x_{i+1} from $X_{i+1} = x_{i+1}P$, $Y_{i+1} = x_{i+1}P_{pub}$, or even from $(w_k^{i+1})_{1\leq k\leq i+1}$, then E_i can forge the proven unforgeable, Boneh-Franklin [3] and Gentry-Silverberg [6] signatures, under the Computational Diffie-Hellman, Bilinear Diffie-Hellman and elliptic-curve discrete logarithm problems hardness assumptions (refer to section 4.2).

5.2.2 Verifiability

Since all the delegation warrants include the identity and public key of the delegator or original signer in addition to the delegation restrictions to be enforced, the verifier of a proxy signature can check its authenticity and then verify whether the signed message comply or not with all the delegation restrictions stated within the warrants set. If so, the verifier can be convinced of the agreement of the original signer and the intermediate proxy signers within the delegation chain on the signed message. In fact as one can see in equation 3, the value of V_t includes the private keys of the original signer and all the intermediate proxy signers in addition to hashs of the all involved delegation warrants. These hashes in conjunction to other public informations are used within the proxy signature verification equation to check the authenticity of the proxy signature.

5.2.3 Strong identifiability

In our scheme, the identity of the proxy signer is tied up with any proxy signature he generates. In fact, any signature over a message produced by a proxy signer E_t includes a warrant m_{t-1} in which his identity is stated as being a designated proxy signer in addition to the hash $h_t = H_2(m, Q_t)$ that ties this latter identity to the message being signed.

5.2.4 Strong undeniability

Considering the strong unforgeability and the strong identifiability properties, only the proxy signer, whose identity is identified within a proxy signature, can generate such a proxy signature. Therefore, he cannot deny his responsibility.

5.2.5 Prevention of misuse

In our scheme, the delegation signature and proxy signing key cannot be used to learn any of the involved parties private keys. Thereby, they cannot be used for other purposes than proxy signing and delegating signing capability. The compliance of these actions with their respective warrants are to be enforced by any entity verifying a proxy signature or being delegated a signing capability on behalf of some other entities.

5.3 Performance evaluation

Pairings are expensive compared to summation and exponentiation. While computing the value of r, the verifier have to compute $n_t = \frac{t(t+1)}{2} + 2$ pairings. This number can be reduced by doing some optimizations in the way r is computed (for the mathematical correctness proof for this optimized form, interested readers are kindly referred to appendix B). In fact, the verifier can compute r as follows:

$$r = \frac{e(S,P) \cdot e(h_A Q_A + U_A, Y_A)^{-h \prod_{j=1}^{t} h_j}}{\prod_{k=1}^{t} e\left(\sum_{i=k}^{t} (\prod_{j=i}^{t} h_j) w_k^i, X_{k-1}\right)^h} \tag{7}$$

Thus, reducing the number of computed pairings to $n_t = t + 2$.

Using the PBC [18] library we implemented the proposed scheme and measured its computational and communication[11] cost. In fact, figure 6 depicts the computational time required for the *proxy signing* procedure, the *proxy signature verification* procedure, and the *optimized proxy signature verification* procedure with respect to the length of the proxy chain. Figure 5, illustrates the computational time required to form a delegation chain and to carry out the signature delegation over it with respect to the intended chain length.

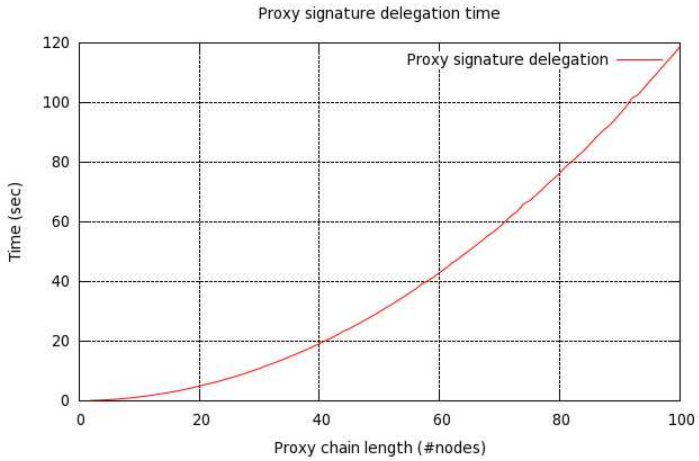


Figure 5: Signature delegation time over the proxy chain

As predicted, we can see that the proxy signature verification using the optimized form of r requires less time than the non optimized one. This is due to the fact that the non-optimized form uses $n_t = \frac{t(t+1)}{2} + 2$ pairings , while the optimized one needs to perform $n_t = t + 2$ pairings.

For instance,as illustrated in figure 6, when the proxy chain is 40 nodes long, the proxy signing procedure takes 0.977 seconds, the non-optimized signature verification procedure requires 7.704 seconds to complete, whereas the optimized proxy signature verification takes 6.379 seconds. In this case, it takes

[11] with respect to the size of generated signature

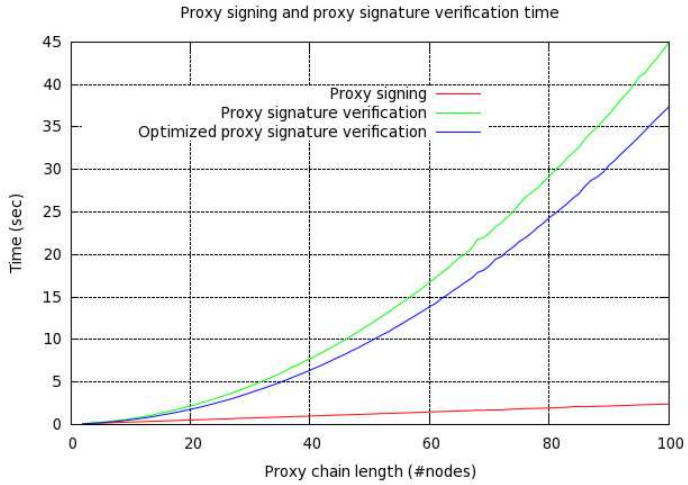


Figure 6: Proxy signing and verification time

19.111 seconds to form the delegation chain of 40 nodes and carry out the signature delegation procedure. This can be more enhanced by applying efficient pairings [2].

Figure 7: Proxy signature size (without including warrants)

Figures 7 depicts the evolution of the proxy signature size (in kilo bytes) with respect to length of the delegation chain. For example, a user at the bottom of delegation chain comprised of 40 nodes will output a proxy signature around 116 kilobyte (without taking into consideration the warrants) in size. If we consider that the size of each of the elements composing the proxy signature, like X_i, Y_i, w_k^i, U_A, h, and S are roughly equal (128 bytes in our benchmark), we can infer that the size of the proxy signature σ_{proxy_t} (having t nodes in the delegation chain) is proportional to $\frac{t^2+5t}{2} + 5$.

In our scheme, computation can be saved by the verifier when checking the authenticity of a proxy signature over a new message from the same proxy signer. In fact, when a proxy signer outputs a signature over a message on behalf of the same proxy chain (assuming that the warrants still permits such an operation: for example delegation's validity period haven't yet expired), only the elements h, h_t, R and S will change. Then, the proxy signer needs to send only the new values of h and S to the

verifier (h_t can be computed by the verifier), thereby, reducing the signature size (to 148 bytes in our benchmark), in the first place. The verifier, can save the following quantities, $K_{deleg_{t-1}}$ and K_{proxy_t}, when checking a signature from the proxy signer at the first time:

$$K_{deleg_{t-1}} = \frac{e(h_A Q_A + U_A, Y_A)^{-\prod_{j=1}^{t-1} h_j}}{\prod_{k=1}^{t-1} e\left(\sum_{i=k}^{t-1}(\prod_{j=i}^{t-1} h_j) w_k^i, X_{k-1}\right)} \quad (8)$$

$$= \frac{e(h_A Q_A + U_A, Y_A)^{-\prod_{j=1}^{t-1} h_j}}{\prod_{i=1}^{t-1}\prod_{k=1}^{i} e\left(w_k^i, X_{k-1}\right)^{\prod_{j=i}^{t-1} h_j}} \quad (9)$$

$$K_{proxy_t} = \frac{1}{\prod_{k=1}^{t} e(w_k^t, X_{k-1})} \quad (10)$$

and use them as a pre-computed values[12] for further signatures verification from the same proxy signer as follows:

$$r = e(S, P) \cdot \left(K_{proxy_t}^{h_t} K_{deleg_{t-1}}\right)^{h h_t} \quad (11)$$

Thus, reducing the computational cost (to 0.009 seconds in our benchmark), in the second place.

6 Conclusion

In this paper we presented a new cryptographic scheme, namely Certificateless Identity-Based Proxy Signature. In this proposed scheme, the signing capability of an original signer, which can be an Attribute Authority within a Grid-computing VO, is delegated to a chain of entities or chain of proxy signers. The used signing delegation in our proposal is warrant-based, thus, making it straightforward to carry user's attributes and also implies fine grained delegation. In fact, a proxy signer within the proxy chain, by just proxy-signing a message or a job request, can claim to be a trusted and legitimate proxy of the corresponding authenticated user, if the presented proxy signature is accepted by the requested resource authorization manager. In this case, the warrant-carried attributes, will be assumed as authentically asserted by the VO Attribute Authority, and can be used for authorization decision making. In addition, being able to output a valid proxy signature implies the authentic delegation relationship between the original signer and the proxy signer. Thereby, resource access manager can be freed from authenticating users by leaving this procedure to be carried out and enforced by a trusted Attribute Authority using the existing and available authentication mechanisms in place when asserting users attributes and delegating his signing capability to them. Using the pairing based library, we implemented the propose scheme and measured its computational cost and bandwidth (signature size) overhead. Our results, shows a satisfying overhead up to a proxy chain of 40 nodes long. Finally, we analyzed the correctness and security of our proposed scheme and showed that it was secure according to the literature security requirements for proxy signing.

References

[1] Sattam S. Al-riyami, Kenneth G. Paterson, and Royal Holloway. Certificateless public key cryptography. In *Advances in Cryptology - ASIACRYPT 2003*, pages 452–473. Springer-Verlag, 2003.

[2] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[12] When validating the proxy signature using the optimized verification form, $K_{deleg_{t-1}}$ can be computed using the formula 8 otherwise formula 9 should be used

[3] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.

[4] Liqun Chen, Hoon Wei Lim, and Wenbo Mao. User-friendly grid security architecture and protocols. In *Security Protocols, 13th International Workshop, Revised Selected Papers*, volume 4631 of *Lecture Notes in Computer Science*, pages 139–156, Cambridge, UK, April 2007. Springer.

[5] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, New York, NY, USA, 1998. ACM.

[6] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 548–566, London, UK, 2002. Springer-Verlag.

[7] Xiaoqin Huang, Lin Chen, Linpeng Huang, and Minglu Li. An identity-based grid security infrastructure model. In *Information Security Practice and Experience, First International Conference, ISPEC 2005, Proceedings*, volume 3439 of *Lecture Notes in Computer Science*, pages 314–325, Singapore, April 2005. Springer.

[8] Mohamed Amin Jabri and Satoshi Matsuoka. Authorization within Grid-Computing Using Certificateless Identity-Based Proxy Signature. HPDC '10: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, June 2010. short paper accepted.

[9] B Lee, H Kim, and K Kim. Strong proxy signature and its applications. In *Proceeding of the 2001 Symposium on Cryptography and Information Security*, pages 603–608, 2001.

[10] Hongwei Li and Shixin Sun. Identity-based cryptography for grid. *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, 2:132–137, 30 2007-Aug. 1 2007.

[11] X. Li, K. Chen, and L. Sun. Certificateless signature and proxy signature schemes from bilinear pairings. *Lithuanian Mathematical Journal*, 45(1):76 – 83, 2005.

[12] Hoon W. Lim and Matthew J. B. Robshaw. On identity-based cryptography and grid computing. In *Computational Science - ICCS 2004, 4th International Conference, Proceedings, Part I*, volume 3036 of *Lecture Notes in Computer Science*, pages 474–477, Kraków, Poland, June 2004. Springer.

[13] Hoon Wei Lim and K.G. Paterson. Identity-based cryptography for grid security. In *e-Science and Grid Computing, 2005. First International Conference on*, pages 10 pp.–404, July 2005.

[14] Hoon Wei Lim and Matthew J. B. Robshaw. A dynamic key infrastructure for grid. In *Advances in Grid Computing - EGC 2005, European Grid Conference, Revised Selected Papers*, volume 3470 of *Lecture Notes in Computer Science*, pages 255–264, Amsterdam, The Netherlands, February 2005. Springer.

[15] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures: Delegation of the power to sign messages (special section on information theory and its applications). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 79(9):1338–1354, 1996.

[16] Wenbo Mao and Wenbo Mao. An identity-based non-interactive authentication framework for computational grids. Technical report, HP Labs Bristol, Trusted Systems Laboratory, Technical Report, 2004.

[17] Jason Novotny, Steven Tuecke, and Von Welch. An online credential repository for the grid: Myproxy. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pages 104–111, Washington, DC, USA, 2001. IEEE Computer Society.

[18] The pairing-based cryptography (pbc) library. [Online]. Available: http://crypto.stanford.edu/pbc/.

[19] Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng. Security analysis of some proxy signatures. In *in Proc. of Information Security and Cryptology (LNCS 2971)*, pages 305–319. Springer-Verlag, 2003.

[20] Wenbo Zhang, Hongqi Zhang, Bin Zhang, and Yan Yang. An identity-based authentication model for multi-domain in grid environment. *Computer Science and Software Engineering, 2008 International Conference on*, 3:165–169, Dec. 2008.

A Proof of the correcteness claim

Let $(P_i)_{1\leq i\leq t}$ and P_A be a set of correctly computed public keys, and $\sigma_{Proxy_t} = \langle m_A, U_A, (m_i)_{1\leq i\leq t-1}, (W_i)_{1\leq i\leq t}, (P_i)_{1\leq i\leq t}, h, S\rangle$ be a valid proxy signature over a given message m, which was produced following our proposed scheme.

Proof. Using the bilinearity property of the pairing e, we have for all $1 \leq i \leq t$:

$$\begin{aligned} e(Y_i, P) &= e(x_i P_{pub}, P) \\ &= e(x_i x_0 P, P) \\ &= e(x_i P, x_0 P) \\ &= e(X_i, P_{pub}) \end{aligned}$$

Similarly, we can proove that $e(Y_A, P) = e(X_A, P_{pub})$.

On the other hand, using the definition of the proxy signing key S_{proxy_t}, we deduce:

$$\begin{aligned} e(S, P) &= e(hh_t S_{Proxy_t} + aP, P) \\ &= e(hh_t S_{Proxy_t}, P) \cdot e(aP, P) \\ &= r \cdot e(hh_t(V_{t-1} + r_t S_t), P) \\ &= r \cdot e(hh_t r_t S_t, P) \cdot e(hh_t V_{t-1}, P) \end{aligned} \tag{12}$$

Using the definition of a private key S_t (see equation 1), we induce the following:

$$\begin{aligned} e(hh_t r_t S_t, P) &= e(hh_t r_t \sum_{k=1}^{t} x_t x_{k-1} Q_k, P) \\ &= e(\sum_{k=1}^{t} hh_t r_t x_t x_{k-1} Q_k, P) \\ &= \prod_{k=1}^{t} e(hh_t r_t x_t x_{k-1} Q_k, P) \\ &= \prod_{k=1}^{t} e(hh_t r_t x_t Q_k, x_{k-1} P) \\ &= \prod_{k=1}^{t} e(h_t w_k^t, X_{k-1})^h \end{aligned} \tag{13}$$

Using equation 1, 2 and 3 we have:

$$\begin{aligned} e(hh_tV_{t-1},P) &= e(hh_t\sum_{i=1}^{t-1}\left(\prod_{j=i}^{t-1}h_j\right)r_iS_i + \\ &\quad hh_t\left(\prod_{j=1}^{t-1}h_j\right)(h_A+r_A)S_A,P) \\ &= \frac{e(\sum_{i=1}^{t-1}hh_t\left(\prod_{j=i}^{t-1}h_j\right)r_iS_i,P)}{e(h\left(\prod_{j=1}^{t}h_j\right)(h_A+r_A)S_A,P)^{-1}} \\ &= \frac{\prod_{i=1}^{t-1}e(r_iS_i,P)^{h\prod_{j=i}^{t}h_j}}{e((h_A+r_A)S_A,P)^{-h\prod_{j=1}^{t}h_j}} \\ &= \frac{\prod_{i=1}^{t-1}e(\sum_{k=1}^{i}r_ix_ix_{k-1}Q_k,P)^{h\prod_{j=i}^{t}h_j}}{e((h_A+r_A)x_Ax_0Q_A,P)^{-h\prod_{j=1}^{t}h_j}} \\ &= \frac{\prod_{i=1}^{t-1}\prod_{k=1}^{i}e(x_{k-1}w_k^i,P)^{h\prod_{j=i}^{t}h_j}}{e((h_A+r_A)Q_A,x_Ax_0P)^{-h\prod_{j=1}^{t}h_j}} \\ &= \frac{\prod_{i=1}^{t-1}\prod_{k=1}^{i}e(w_k^i,x_{k-1}P)^{h\prod_{j=i}^{t}h_j}}{e(h_AQ_A+U_A,Y_A)^{-h\prod_{j=1}^{t}h_j}} \\ &= \frac{\prod_{i=1}^{t-1}\prod_{k=1}^{i}e(w_k^i,X_{k-1})^{h\prod_{j=i}^{t}h_j}}{e(h_AQ_A+U_A,Y_A)^{-h\prod_{j=1}^{t}h_j}} \end{aligned} \tag{14}$$

Finally, from equations 12, 13, and 14 we can deduce:

$$r = \frac{e(S,P)\cdot e(h_AQ_A+U_A,Y_A)^{-h\prod_{j=1}^{t}h_j}}{\prod_{i=1}^{t}\prod_{k=1}^{i}e(w_k^i,X_{k-1})^{h\prod_{j=i}^{t}h_j}} \tag{15}$$

which concludes our scheme's correcteness proof □

B Optimized proxy signature correctness proof

Proof. Given the fact that the proposed scheme is correct (refer to Claim 1) we need only to show that the following equation holds:

$$r = \frac{e(S,P)\cdot e(h_AQ_A+U_A,Y_A)^{-h\prod_{j=1}^{t}h_j}}{\prod_{i=1}^{t}\prod_{k=1}^{i}e(w_k^i,X_{k-1})^{h\prod_{j=i}^{t}h_j}} = \frac{e(S,P)\cdot e(h_AQ_A+U_A,Y_A)^{-h\prod_{j=1}^{t}h_j}}{\prod_{k=1}^{t}e\left(\sum_{i=k}^{t}(\prod_{j=i}^{t}h_j)w_k^i,X_{k-1}\right)^h}$$

Claim 2. *For any given $t\in\mathbb{N}^*$, the following equation holds:*

$$\alpha_t = \sum_{i=1}^{t}\sum_{k=1}^{i}r_ix_i\left(\prod_{j=i}^{t}h_j\right)x_{k-1}Q_k = \sum_{k=1}^{t}\sum_{i=k}^{t}r_ix_i\left(\prod_{j=i}^{t}h_j\right)x_{k-1}Q_k \tag{16}$$

Proof. Let $t=1$, then:

$$\begin{aligned} \alpha_1 &= \sum_{i=1}^{1}\sum_{k=1}^{i}r_ix_i\left(\prod_{j=i}^{1}h_j\right)x_{k-1}Q_k = r_1x_1h_1x_0Q_1 \\ &= \sum_{k=1}^{1}\sum_{i=k}^{1}r_ix_i\left(\prod_{j=i}^{1}h_j\right)x_{k-1}Q_k \end{aligned}$$

So statement 16 holds for $t = 1$.

Assume that for a given $n \in \mathbb{N}^*$, statement 16 holds for $t = n$; that is:

$$\alpha_n = \sum_{i=1}^{n}\sum_{k=1}^{i} r_i x_i \left(\prod_{j=i}^{n} h_j\right) x_{k-1}Q_k = \sum_{k=1}^{n}\sum_{i=k}^{n} r_i x_i \left(\prod_{j=i}^{n} h_j\right) x_{k-1}Q_k$$

Let $t = n + 1$

$$\begin{aligned}
\alpha_{n+1} &= \sum_{i=1}^{n+1}\sum_{k=1}^{i} r_i x_i \left(\prod_{j=i}^{n+1} h_j\right) x_{k-1}Q_k = \sum_{i=1}^{n}\sum_{k=1}^{i} r_i x_i \left(\prod_{j=i}^{n+1} h_j\right) x_{k-1}Q_k + \sum_{k=1}^{n+1} r_{n+1}x_{n+1}h_{n+1}x_{k-1}Q_k \\
&= h_{n+1}\sum_{i=1}^{n}\sum_{k=1}^{i} r_i x_i \left(\prod_{j=i}^{n} h_j\right) x_{k-1}Q_k + \sum_{k=1}^{n} r_{n+1}x_{n+1}h_{n+1}x_{k-1}Q_k + r_{n+1}x_{n+1}h_{n+1}x_nQ_{n+1} \\
&= h_{n+1}\sum_{k=1}^{n}\sum_{i=k}^{n} r_i x_i \left(\prod_{j=i}^{n} h_j\right) x_{k-1}Q_k + \sum_{k=1}^{n} r_{n+1}x_{n+1}h_{n+1}x_{k-1}Q_k + r_{n+1}x_{n+1}h_{n+1}x_nQ_{n+1} \\
&= \sum_{k=1}^{n}\sum_{i=k}^{n} r_i x_i \left(\prod_{j=i}^{n+1} h_j\right) x_{k-1}Q_k + \sum_{k=1}^{n} r_{n+1}x_{n+1}h_{n+1}x_{k-1}Q_k + r_{n+1}x_{n+1}h_{n+1}x_nQ_{n+1} \\
&= \sum_{k=1}^{n}\left(\sum_{i=k}^{n} r_i x_i \left(\prod_{j=i}^{n+1} h_j\right) x_{k-1}Q_k + r_{n+1}x_{n+1}h_{n+1}x_{k-1}Q_k\right) + r_{n+1}x_{n+1}h_{n+1}x_nQ_{n+1} \\
&= \sum_{k=1}^{n}\sum_{i=k}^{n+1} r_i x_i \left(\prod_{j=i}^{n+1} h_j\right) x_{k-1}Q_k + r_{n+1}x_{n+1}h_{n+1}x_nQ_{n+1} \\
&= \sum_{k=1}^{n+1}\sum_{i=k}^{n+1} r_i x_i \left(\prod_{j=i}^{n+1} h_j\right) x_{k-1}Q_k
\end{aligned}$$

Then statement 16 holds for $t = n + 1$. □

Now, we need to prove that:

$$\prod_{i=1}^{t}\prod_{k=1}^{i} e(w_k^i, X_{k-1})^{h\prod_{j=i}^{t} h_j} = \prod_{k=1}^{t} e\left(\sum_{i=k}^{t}(\prod_{j=i}^{t} h_j)w_k^i, X_{k-1}\right)^h \tag{17}$$

Using the bilinearity property of a bilinear pairing (map), we have:

$$\begin{aligned}
\prod_{k=1}^{t} e\left(\sum_{i=k}^{t}\left(\prod_{j=i}^{t} h_j\right) w_k^i, X_{k-1}\right)^h &= \prod_{k=1}^{t} e\left(\sum_{i=k}^{t} r_i x_i \left(\prod_{j=i}^{t} h_j\right) Q_k, x_{k-1}P\right)^h \\
&= \prod_{k=1}^{t} e\left(\sum_{i=k}^{t} r_i x_i \left(\prod_{j=i}^{t} h_j\right) x_{k-1}Q_k, P\right)^h \\
&= e\left(\sum_{k=1}^{t}\sum_{i=k}^{t} r_i x_i \left(\prod_{j=i}^{t} h_j\right) x_{k-1}Q_k, P\right)^h
\end{aligned}$$

Using Claim 2, we conclude that:

$$\begin{aligned}
\prod_{k=1}^{t} e\left(\sum_{i=k}^{t}\left(\prod_{j=i}^{t} h_j\right) w_k^i, X_{k-1}\right)^h &= e\left(\sum_{i=1}^{t}\sum_{k=1}^{i} r_i x_i \left(\prod_{j=i}^{t} h_j\right) x_{k-1}Q_k, P\right)^h \\
&= \prod_{i=1}^{t} e\left(\sum_{k=1}^{i} r_i x_i \left(\prod_{j=i}^{t} h_j\right) x_{k-1}Q_k, P\right)^h \\
&= \prod_{i=1}^{t}\prod_{k=1}^{i} e\left(r_i x_i \left(\prod_{j=i}^{t} h_j\right) x_{k-1}Q_k, P\right)^h \\
&= \prod_{i=1}^{t}\prod_{k=1}^{i} e(r_i x_i Q_k, x_{k-1}P)^{h\prod_{j=i}^{t} h_j} \\
&= \prod_{i=1}^{t}\prod_{k=1}^{i} e(w_k^i, X_{k-1}P)^{h\prod_{j=i}^{t} h_j}
\end{aligned}$$

Then statement 17 holds. Thereby, the following equation also holds:

$$r = \frac{e(S,P)\cdot e(h_A Q_A + U_A, Y_A)^{-h\prod_{j=1}^{t} h_j}}{\prod_{i=1}^{t}\prod_{k=1}^{i} e(w_k^i, X_{k-1})^{h\prod_{j=i}^{t} h_j}} = \frac{e(S,P)\cdot e(h_A Q_A + U_A, Y_A)^{-h\prod_{j=1}^{t} h_j}}{\prod_{k=1}^{t} e\left(\sum_{i=k}^{t}(\prod_{j=i}^{t} h_j) w_k^i, X_{k-1}\right)^h}$$

Which concludes our proof. □