Trans-Organizational Role-Based Access Control in Android

Secure Mechanism for Verifying User-Role Assignments of Organizations

Jason Paul Cruz and Yuichi Kaji
Nara Institute of Science and Technology, Graduate School of Information Science
Nara, Japan
E-mail: jpmcruz@ymail.com, kaji@is.naist.jp

***Abstract*—The role-based access control (RBAC) is a natural and versatile model of the access control principle. In the real world, it is common that an organization provides a service to a user who owns a certain role that was issued by a different organization. However, such a trans-organizational RBAC is not common in a computer network because it is difficult to establish both the security that prohibits malicious impersonation of roles and the flexibility that allows small organizations/individual users to fully control their own roles. To solve this problem, this study proposes a mechanism that makes use of the hierarchical ID-based encryption scheme and the challenge-response authentication protocol. The proposed mechanism contributes to achieve both the security and the flexibility and it provides additional features that are common in physical communication but are not obvious in the cyberworld. This study also reports a prototyping system that is implemented on Android-enabled mobile devices. The proposed system employs the needed cryptographic mechanisms, and new technologies, namely, near-field communication and two-dimensional codes, are employed to realize locally closed communication between devices.**

Keywords-role-based access control; trans-organizational role; information security; ID-based encryption; service coalition; Android.

I. INTRODUCTION

The role-based access control (RBAC) [1] is a widely accepted framework that describes the access control relation among users and services. In RBAC, users are associated with roles, and roles are associated with services. This framework is compatible with the access control requirements of real-world organizations and is employed in the computer systems of many organizations/companies. However, it must be noted that RBAC is a versatile framework, and roles are often used in a trans-organizational manner. For example, students are often allowed to be admitted to a museum with discounted admission fee. In this example, the "student" role that is issued by an organization (school) is used by another organization (museum) to determine if a guest is eligible to receive a certain service (discounted admission). This kind of trans-organizational use of roles is, unfortunately, not common in computer networks. Even if one has a certain role that is issued by an organization, there is no way to convince a third-party organization that he/she really has that role.

To realize a trans-organizational RBAC mechanism in a computer network, two issues should be considered; the security and the flexibility. With regard to security, the mechanism should prevent malicious users from disguising their roles. This requirement is naturally accomplished in real-world services with the use of physical certificates, such as passports and ID-cards, which are difficult to forge or alter. This problem, however, is not obvious in a computer system. Digital certificates [2] can be utilized as an analogue of physical certificates, but the use of digital certificates is not favorable from the viewpoint of realization cost, which can discourage small companies and non-profit organizations from participating in the framework. Another less sophisticated approach to the security problem is to let a service-providing organization (the museum in the above example) inquire a role-issuing organization (school) about the user-role assignment. This approach works fine in some cases [3], but a focal point of this approach is the necessity for the agreed beneficial relationship among organizations. Consequently, it is difficult for a new organization to join the partnership, severely restricting the trans-organizational utilization of roles.

The current study aims to develop a practical mechanism that realizes the trans-organizational utilization of roles. First, we extend the model of RBAC to represent the trans-organizational usage of roles. This simple extension clarifies the components and requirements that are needed in the framework of trans-organizational RBAC. Then, we investigate a realization of a user-role assignment that is secure (users cannot disguise roles), user-oriented (users can disclose their roles to any organization), and open (anyone can verify if a user has a certain role that is managed and issued by another organization). The crucial point of this realization is to make use of hierarchical ID-based encryption (HIBE) [4][5], which allows an arbitrary string to be used as a public encryption key. Our key idea is to define correspondence between the roles and keys of HIBE and to employ a challenge-response authentication protocol that will be used for verifying if a user really has an asserted role. The hierarchical nature of HIBE makes our scheme suitable for the trans-organizational utilization of roles, and furthermore, allows flexible role management operations, such as the endorsement and delegation of roles. A prototype system of the proposed trans-organizational RBAC will also be introduced. For the usability of the trans-organizational RBAC, it is highly desirable for a user to be able to carry

 ISBN: 978-1-61208-365-0

his/her roles all the time. To verify the practicality of the proposed scheme, we implemented the proposed system on Android-enabled mobile devices. The implementation contains the realization of cryptographic functions that are essential for handling cryptographic keys and the development of a scheme that allows two devices to perform the challenge-response authentication by utilizing local and closed communication. The prototype demonstrates that the proposed scheme is simple, lightweight, and completely practical for realizing the trans-organizational RBAC. The rest of this paper is organized as follows. Section II introduces the RBAC and the different models associated with it. Section III discusses the technical aspects of HIBE. Section IV presents the structure, procedures, and features of the proposed framework. Section V discusses the realization and implementation of the proposed system in Android-enabled devices. Section VI provides the conclusion and future work.

II. MODELS FOR THE ROLE-BASED ACCESS CONTROL

In the simplest model of the RBAC [1], the access structure is defined by three sets and two relations. In this paper, we use U for the set of *users*, R for the set of *roles*, and S for the set of *services*. A *user-role assignment*, UA, is a subset of $U \times R$, and a *role-service assignment*, SA, is a subset of $R \times S$. A user u is eligible to access a service s if and only if there is a role r such that $(u, r) \in$ UA and $(r, s) \in$ SA. The access control should be made in such a way that services are provided to eligible users only. In general, the user-role assignment UA is defined by an entity that issues roles in R, and the role-service assignment SA is defined by an entity that provides the services in S. In this paper, the former is called a *role-issuing entity*, and the latter is called a *service-providing entity*. If RBAC is utilized in a single organization, then we can regard that the role-issuing entity and the service-providing entity are the same identical organization, and that the service-providing entity should have no difficulty referring to the user-role assignment. In this case, the eligibility of a user u to a service s can be easily determined.

On the other hand, in the real world, there are many cases wherein a service-providing entity is a different organization from a role-issuing entity. As stated in the previous section, a school issues the "student" role to its students, and an external organization, such as a museum, provides services to users who hold the "student" role. In this case, the service-providing organization (museum) is a completely independent organization from the role-issuing organization (school), and the service-providing organization is not expected to refer to the user-role assignment that was defined by the role-issuing organization. To discuss such a situation, we first consider an extended model of RBAC.

The *trans-organizational RBAC* is defined similarly to the usual RBAC, but a set O of *organizations* is defined in addition to the sets of users, roles, and services. Furthermore, the set R of roles is partitioned into several subsets, with each subset of R associated with an element in O, that is, $R = R_{o_1} \cup \cdots \cup R_{o_n}$, where $o_1, \ldots, o_n \in O$ and $o_i \cap o_j = \emptyset$ if $i \neq j$. To make the relation among roles and organizations explicit, a role r in R_{o_i} is written as $o_i.r$. Similarly, the user-role assignment UA is partitioned into disjoint subsets; $\mathrm{UA} = \mathrm{UA}_{o_1} \cup \cdots \cup \mathrm{UA}_{o_n}$, where $\mathrm{UA}_{o_i} \subset U \times R_{o_i}$. Obviously, $o_i.r \in R_{o_i}$ means that the role $o_i.r$ is managed by the organization o_i, and the assignment of users to $o_i.r$ is fully controlled by that organization o_i. In the trans-organizational RBAC, upon a request from a user u to a service s, the service-providing organization needs to check if there is an organization $o_i \in O$ and a role $o_i.r \in R_{o_i}$ such that $(u, o_i.r) \in \mathrm{UA}_{o_i}$ and $(o_i.r, s) \in$ SA. Assuming that the user u declares the role $o_i.r$ to utilize, then all the service-providing organization needs to do is check if $(u, o_i.r) \in \mathrm{UA}_{o_i}$ or not. However, it should be noted that the role-issuing organization o_i can be a different organization from the service-providing organization in general. The confirmation of $(u, o_i.r) \in \mathrm{UA}_{o_i}$, which is sometimes called an *authentication*, is not as obvious for the service-providing organization as in the single-organization case. If the confirmation cannot be established, then a malicious user may try to access a service by asserting a role that the user does not actually have.

It is essential in the trans-organizational RBAC to realize a secure authentication mechanism, and this problem can be solved using two approaches. The first approach is to utilize digital certificates that are protected by the digital signatures of the role-issuing entities. This kind of certificate is sometimes called an attribute certificate [2] and is regarded as a digitalized version of physical certificates, such as ID-cards. The problem in this approach is the maintenance cost of the public-key infrastructure (PKI) [6][7]. Different from written signatures, continuous efforts are essential to keep digital signatures secure and functional. PKI is widely recognized as expensive, and this cost issue prevents small organizations from participating in a PKI-based framework. The second, rather political, approach to the authentication problem is to arrange a mutual agreement between role-issuing organizations and service-providing organizations. If several organizations share an identical benefit, then they can set up a partnership and mutually disclose their user-role assignments. A good example of this approach can be found in the Shibboleth project [3], but we need to remark that this framework is essentially a semi-closed one. An organization will not be allowed to join the partnership if that organization cannot offer recognizable benefits to the organizations involved, consequently limiting the trans-organizational utilization of roles.

III. HIERARCHICAL ID-BASED ENCRYPTION

A public-key encryption is a cryptography that utilizes two different keys for encryption and decryption. In a typical public-key encryption, such as RSA [8], a user sets up his/her key pair by himself/herself. One of the keys in the key pair is called an encryption key and is disclosed to the public. The other key is called a decryption key and is kept secretly by the user. In many cases, the keys are constructed from randomly selected information, which means that the

 ISBN: 978-1-61208-365-0

keys look like random data. A separate mechanism, such as PKI, is needed to associate public encryption keys with users. However, PKI makes the system complicated and costly [6][7].

An ID-based encryption [4] is a special public-key encryption. Different from the usual public-key encryption, a user first chooses his/her encryption key. An interesting point in the ID-based encryption is that, under an appropriate setting, one's identity, such as an e-mail address, can be used as an encryption key. After choosing the encryption key, the user submits the encryption key to a trusted authority which we call a *key generator*. The key generator examines the eligibility of the user and then, upon confirmation of the user's eligibility, computes the decryption key that corresponds to the submitted encryption key. Different from the usual public-key encryption, the correspondence between users and encryption keys becomes obvious if the identities of users are chosen as the encryption keys. Consequently, the costly mechanism of PKI is not needed in the framework of ID-based encryptions [4].

The HIBE [5] is an extension of the ID-based encryption wherein identities and functions of the key generator are realized in a hierarchical manner. In this paper, we write a hierarchical identity (abbreviated simply as ID) by a sequence of strings $S = s_1.s_2.\ldots.s_n$, where n is a non-negative integer called a *level* of S, and s_i with $1 \leq i \leq n$ is a string. If an ID $S = s_1.s_2.\ldots.s_n$ is a prefix of $S' = s'_1.s'_2.\ldots.s'_{n'}$, then we say that S is a *super-ID* of S' and S' is a *sub-ID* of S. In HIBE, an ID can be regarded as an encryption key by itself, although, it is sometimes convenient to distinguish IDs from encryption keys explicitly. In the following discussion, we write ek_S and dk_S for the encryption and decryption keys that correspond to the identity S, respectively. In the original ID-based encryption, all decryption keys are solely generated by a trusted key generator. In HIBE, however, the generation of decryption keys is made in a hierarchical manner; the key pair $(\mathrm{ek}_S, \mathrm{dk}_S)$ for a level-one ID $S = s_1$ is generated by a designated key generator which we call a *root key generator* and is issued to an appropriate user who is eligible to hold the key pair. A user who has a key pair $(\mathrm{ek}_S, \mathrm{dk}_S)$ for an ID S can generate a key pair $(\mathrm{ek}_{S'}, \mathrm{dk}_{S'})$ for an ID S' that is a sub-ID of S. The functions used in HIBE are described below. There is complicated mathematics behind these functions, but we omit them because they are not the subject of this study. The main body of HIBE consists of the following four procedures:

Initialize() is a procedure that is executed by the root key generator in the initialization of the HIBE system. This procedure determines the public and secret parameters used in the system. The secret parameter is kept secretly by the root key generator, and the public parameter is disclosed to the public. The value of the public parameter is used in the following three procedures, although we do not write them explicitly in the notation for simplicity.

KeyGenerate(S, $(\mathrm{ek}_{S'}, \mathrm{dk}_{S'})$) is a procedure that computes the decryption key for the given ID S. More precisely, the procedure generates dk_S if S is a sub-ID of S' and $\mathrm{dk}_{S'}$ is a correct decryption key of S'. If not, the procedure fails to compute dk_S. We remark that dk_S cannot be computed if one does not know a correct decryption key of a super-ID of S.

Encrypt(k, m) encrypts data m by using k as an encryption key.

Decrypt(k, c) decrypts data c by using k as a decryption key.

If $(\mathrm{ek}_S, \mathrm{dk}_S)$ is a key pair that is correctly generated by KeyGenerate and $c = \mathrm{Encrypt}(\mathrm{ek}_S, m)$ is an encryption of m constructed by using the encryption key ek_S, then $\mathrm{Decrypt}(\mathrm{dk}_S, c)$ returns m.

HIBE is useful when used in a challenge-response authentication protocol. Consider a scenario that involves two people, a *prover* and a *verifier*. The prover asserts himself/herself as a genuine user with an identity S, but the verifier is not sure if this assertion is true or not. In this case, the verifier can determine if the prover is genuine or not by executing the following steps. First, the verifier chooses a random message m, and then encrypts m by using the encryption key ek_S for the asserted ID S. The obtained ciphertext $c = \mathrm{Encrypt}(\mathrm{ek}_S, m)$, which is called a *challenge*, is passed to the prover. The prover is requested to decrypt c by using the decryption key dk_S that a genuine user should possess. If the prover returns m as the result of the decryption, then he/she succeeds to make a correct *response* and is authorized as a user with the identity S. If the response is wrong, then the prover is rejected. Through this challenge-response protocol, the verifier is able to determine if the prover has the ID S. At this point, it should be noted that the ID S is indeed a hierarchical identity; the fact that the prover posseses the decryption key dk_S means that somebody who had a certain super-ID of S authorizes the prover to have the identity S. In other words, with the challenge-response protocol, the prover confirms a "chain of trust" that originates from the root key generator. This scenario is favorable in an open system with many unspecified users.

IV. PROPOSED SCHEME

A. Overview

We now consider an authentication mechanism that is suitable for the trans-organizational utilization of roles. Our idea is to represent roles by hierarchical identities that work as encryption keys of HIBE. For example, if we would like to define a "student" role of A-university, then a hierarchical identity, such as "A-univ.student", is introduced and used as an encryption key of HIBE. Decryption keys are managed so that users with a role r possess the correct decryption key dk_r of r. A service-providing organization can verify if a user has the role r by examining the user by using the challenge-response protocol with r used as an encryption key of HIBE. Note that the service-providing organization does not have to know anything about r beforehand and does not have to make any contract or inquiry to the role-issuing organization that has assigned r to the user because r itself is used as an encryption key. This feature makes it easy to verify the user-role assignment of users even if the role is issued by another organization. In the proposed framework,

 ISBN: 978-1-61208-365-0

there is no essential difference between users and role-issuing organizations because they both receive valid key pairs from superordinate entities and they both have the ability to generate new roles and corresponding key pairs by utilizing their keys in possession. However, for an easy understanding of the proposed framework and for simplicity, we will distinguish users from role-issuing organizations and introduce three component procedures that are needed for defining a user-role assignment. In the following, we extend the hierarchical notions of IDs to roles. If r_1 is a super-ID of another ID r_2, then the role represented by r_1 is a *super-role* of the role represented by r_2. The term *sub-role* is defined in the same way.

B. Procedures

Fig. 1 shows the overall structure of the proposed model. In this model, we assume the existence of a designated root key generator that is trusted by all users and organizations. The root key generator executes Initialize() of HIBE and determines the public and secret parameters. The public parameter is disclosed to the public and should be accessible to all users and organizations. The root key generator secures the secret parameter and uses it to generate key pairs for level-one roles.

1) Setting up an organization

An organization o_1 chooses its identity string, say S_{o_1}, and requests the root key generator to approve that the organization uses S_{o_1} as its identity. If the root key generator approves the request, it computes the decryption key dk_{So_1} and sends this key to o_1 by using a secure communication channel. As a result, o_1 possesses a correct key pair $(\mathrm{ek}_{So_1}, \mathrm{dk}_{So_1})$ of its identity S_{o_1}. The organization o_1 then defines the set R_{o_1} of roles it should manage. All roles in R_{o_1} must be sub-roles of S_{o_1}, where the identity S_{o_1} is regarded as a "role". Note that o_1 can compute the decryption key of any role $o_1.r \in R_{o_1}$ by utilizing the function of $\mathrm{KeyGenerate}(o_1.r, (\mathrm{ek}_{So_1}, \mathrm{dk}_{So_1}))$, because o_1 knows the correct key pair $(\mathrm{ek}_{So_1}, \mathrm{dk}_{So_1})$ and $o_1.r$ is a sub-ID of S_{o_1}.

On the other hand, organizations other than o_1 cannot compute the decryption key of the role $o_1.r$ because the trusty key generator does not disclose dk_{So_1} to other organizations. Identities of roles in R_{o_1} can be open to the public, but the corresponding decryption keys must be kept secret by organization o_1.

2) Defining a user-role assignment

To assign a role $o_1.r$ to a user u, the organization o_1 gives the key $\mathrm{dk}_{o_1.r}$ to user u by using a secure communication channel. User u records $\mathrm{dk}_{o_1.r}$ as the decryption key of the role $o_1.r$, and keeps the key secure.

3) Verifying a user-role assignment

Assume that a user u visits a service-providing organization o_2 and asserts that he/she has the role $o_1.r$ that was assigned by the role-issuing organization o_1. The organization o_2 needs to verify if the assertion of user u is true or not. The verification can be done by using the challenge-response protocol; the organization o_2 chooses a random data m and requests user u to decrypt $c = \mathrm{Encrypt}(\mathrm{ek}_{o_1.r}, m)$. Note that we are using HIBE, and the encryption key $\mathrm{ek}_{o_1.r}$ is the same as (or easily derived from) the hierarchical identity $o_1.r$ of the asserted role. If the user really has the role $o_1.r$, then he/she must possess the decryption key $\mathrm{dk}_{o_1.r}$ that is provided by the organization o_1 and should be able to decrypt the challenge c. Remark that the service-providing organization o_2 can verify if the user u holds the role $o_1.r$ without querying the role-issuing organization o_1 and that user u has little chance to disguise his/her role.

C. Managing Roles

1) Personalization of roles

In the proposed framework, the relation between users and roles is represented by the possession of cryptographic keys by users. This approach involves a possible security risk; a leakage of keys. If, for example, a role r is assigned to several users, then all those users have the same key dk_r. If one of those users is unconscious of security, then he/she may let other persons use the key dk_r. Such an inappropriate usage of keys can obstruct fair and reliable access control. To deter such irresponsible behavior of users, a role-issuing organization can "personalize" a role by appending an additional string to the identity of roles. Assume for example that there are several students in A-university. In this case, instead of using a general role, such as "A-univ.student", the university can define personalized roles, such as A-univ.student.Alice and A-univ.student.Bob, and provide

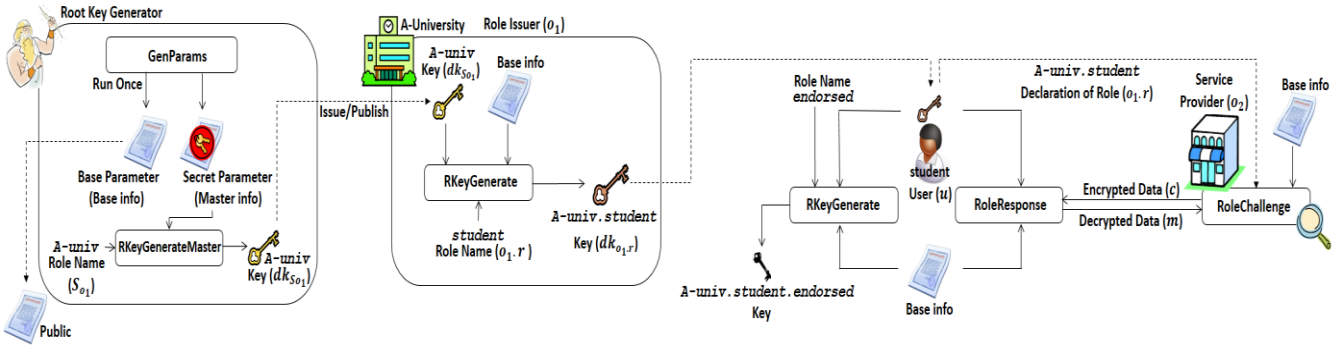


Figure 1. Overview of the proposed structure.

decryption keys of these roles to Alice and Bob, respectively. With this kind of personalization, a student will be more conscious of leaking/losing his/her key to another person because he/she will have the risk of being identified and subsequently punished for irresponsible behavior. The theft/loss of keys remains a risk, but such risk also exists for ID-cards used in the real world. We cannot say that the proposed framework is "more secure than" but we may say it is "as secure as" the real-world role management.

2) Hierarchical issuance of roles

In the proposed scheme, there is no essential difference between organizations and users. A user can compute decryption keys from the key that he/she already has and issue a new sub-role of the role that he/she already has. This function can be used to realize some personal activities that are not considered in the conventional RBAC approach. One possible example is the endorsement of another person. In the real world, an endorsement among individuals sometimes plays an important role. Semi-closed organizations, such as academic societies and golf clubs, have the tradition or policy that a newcomer must be endorsed or referred by a current member. This mechanism can be realized using the proposed scheme. Consider for example that Alice is an authorized member of XYZ golf club and is given a personalized role "XYZ-golf.member.alice" and its corresponding decryption key. If Alice would like to endorse Bob to the club, then she can generate a new sub-role "XYZ-golf.member.alice.endorsed" and its corresponding decryption key. By providing the decryption key to Bob, Bob can demonstrate that he is really endorsed by Alice. Using the HIBE and the challenge-response authentication, the club does not have to inquire Alice for the verification of the endorsement. Besides personal endorsement, we conjecture that a broad range of personal relations can be implemented by utilizing the hierarchical roles.

V. REALIZATION

The proposed scheme was implemented in Android-enabled mobile devices. In the proposed scheme, the user-role assignments are represented by possession of decryption keys by users. A role-issuing organization does not have to construct and maintain large databases for recording the user-role assignments, and it does not have to be bothered by inquiries of other organizations with regard to user-role assignments. The created Android application implements all the functions of the HIBE and the proposed scheme for ease of use and accessibility.

The prototype contains several functions that correspond to components in Fig. 1. The functions of the root key generator mainly consist of two operations: GenParams and RKeyGenerateMaster. GenParams utilizes the Initialize() function of HIBE and generates the public and secret parameters, where the public parameter is disclosed to the public. An organization o_i that would like to participate in this system chooses its identity, say S_{o_i}, and asks the root key generator to compute the decryption key $\mathrm{dk}_{S_{o_i}}$. The root key generator utilizes RKeyGenerateMaster to compute $\mathrm{dk}_{S_{o_i}}$, which needs the information of the secret parameter and hence this function is accessible to the root key generator only. The generated key $\mathrm{dk}_{S_{o_i}}$ is transferred to the organization o_i through general communication means, such as Wi-Fi, Bluetooth, Android Beam, and NFC. The role-issuing organization o_i now has the key pair $(\mathrm{ek}_{S_{o_i}}, \mathrm{dk}_{S_{o_i}})$ for the ID S_{o_i}. By using the function of RKeyGenerate, this key pair, and the public parameter that has been disclosed, the organization o_i can compute valid key pairs of sub-IDs of S_{o_i}. A user receives, possibly many, keys from organizations, each of which corresponds to a role in an organization. The user safely stores these keys in his/her device and accesses them for the RKeyGenerate and RoleResponse functions. RoleResponse provides the function of the prover for the challenge-response authentication and interacts with the RoleChallenge function that is invoked by a service-providing organization.

The cryptographic operations used in these functions are performed using the Java Pairing-Based Cryptography (JPBC) library [9], which is a collection of classes and methods for handling underlying pairing-based cryptosystems. Over JPBC, we implemented the HIBE that was proposed by Gentry [5].

The most complicated but important communication in the proposed scheme is the challenge-response authentication between a user and a service-providing organization. Several messages must be exchanged between two devices, and we provide two different schemes to realize this communication, namely, the use of near-field communication (NFC) and quick-response (QR) codes.

NFC is a contactless technology used to transmit small amounts of data across short distance. NFC has three modes of operation, and this study tackles only P2P mode. NFC messages in Android are handled using the NFC Exchange Format (NDEF). In the proposed Android application, the Intent Filters that listen to the intent action of NfcAdapter.ACTION_NDEF_DISCOVERED were added to the RoleChallenge and RoleResponse activities to be able to receive NFC data [10][11]. Only the MIME type of text/plain was included in the application as we are only concerned with passing and receiving data of type string. Given that at least two activities have the same intent filter that responds to an NFC tap, users are, by default, prompted to select which application in the mobile device to use, making the application tedious to use. To solve this problem, the foreground dispatch system was utilized. The foreground dispatch system is used to make a particular activity have priority over other activities. This allows a particular activity to become the default receiver when it is on the foreground.

QR Code is a type of 2D barcode that is capable of handling different types of data [12]. This code can accommodate high capacity of data in a small area, which is sufficient to include the challenge-response data in one code symbol. The camera hardware of mobile phones can be used as scanners for QR codes generated for the challenge/response actions. For this implementation, the camera hardware of the device was programmed and the ZXing ("zebra crossing") library, which is an open-source library that supports the decoding and generation of

 ISBN: 978-1-61208-365-0

barcodes, was used to obtain the data [13]. This feature allows the interaction of two users without NFC-capable devices and without the Internet.

Typically, the challenge-response authentication is carried out as follows:

1. A user, the prover, opens the application and goes to the "Role Response" option. Then, the prover selects the base file that contains the public parameter and the role file that contains the role he/she wants to use.
2. A service-providing organization, the verifier, opens the application and goes to the "Role Challenge" option. To start the verification process, the verifier selects the same base file and either types the role indicated by the prover or obtains the role automatically via NFC (first tap). Once the role is received, a random challenge data is created.
3. The verifier then sends this challenge data to the prover via NFC (second tap) or by generating a QR code for the prover to scan.
4. After receiving the challenge data via NFC or scanning of the Challenge QR Code, a random response data is calculated and created in the prover's device based on the role file selected.
5. The prover then sends this response data to the verifier, again, via NFC (third tap) or by generating a QR code for the verifier to scan.
6. After receiving the response data via NFC or scanning of the Response QR Code, the Role Challenge indicates if the assumed role is verified or is a mismatch.

Several screen shots of the prototype are shown in Fig. 2. Another possibility for the realization of the user-side system is through the use of smartcards that are compatible with the NFC technology [14].

VI. Conclusion and Future Work

A trans-organizational RBAC is considered and extended to represent the trans-organizational usage of roles. The proposed scheme provides a secure mechanism for verifying the user-role assignments of organizations. The proposed scheme was developed on Android-enabled mobile devices for ease of use and accessibility. Compared to other similar approaches, the proposed scheme provides more flexibility and autonomy while maintaining security. This mechanism allows the realization of many collaborative right managements that are common in physical communication but are difficult to implement over computer networks. Even with the given advantages, the proposed scheme remains subject to the classical issue of compromised secret keys; the proposed scheme is based on the assumption that keys are managed appropriately and protected well. If dk_S is compromised for an unfortunate reason, the keys of the sub-roles of S should be redeployed. This problem can be mitigated by utilizing the personalized and fixed-term roles, but it is encouraged in general to provide more protection for the keys of roles of higher-level. Taking such issue into consideration, future research will focus on the inclusion and integration of expiration dates on the roles. Moreover, the prototype will be expanded to non-Android devices, such as iPhones and Windows mobile devices, for interoperability.

References

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," IEEE Computer, 29, 2, pp. 38–47, 1996.

[2] S. Farrell and R. Housley, "An internet attribute certificate profile or authorization," RFC 3281, 2002.

[3] The Shibboleth System, accesible online: http://shibboleth.internet2.edu/ [accessed: 2014-05-08].

[4] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," Proc. of the Advances in Cryptology (CRYPTO) 2001, pp. 213–229, 2001.

[5] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," Proc. of the Advances in Cryptology (ASIACRYPT) 2002, pp. 548–566, 2002.

[6] C. M. Ellison et al., "SPKI certificate theory," RFC 2693, 1999.

[7] P. Gutmann, "Simplifying public key management," IEEE Computer, 37, 2, pp. 101–103, 2004.

[8] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, 21, 2, pp. 120–126, 1978.

[9] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," Proc. Of the IEEE Symposium on Computers and Communications (ISCC) 2011, pp. 850–855, 2011.

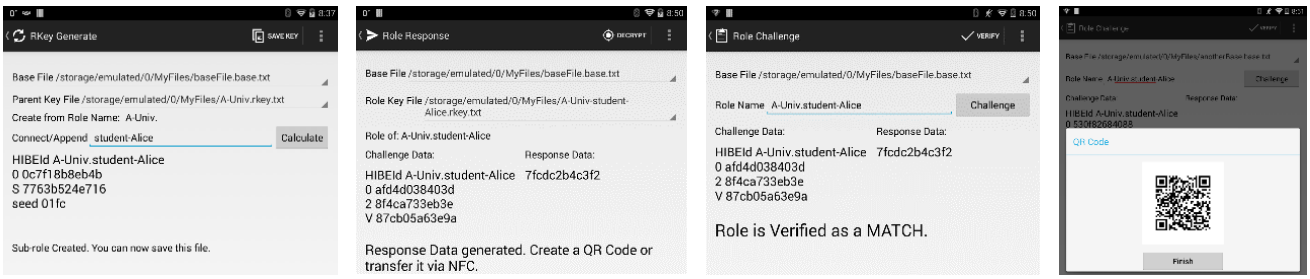
[10] R. Meier, "Professional Android 4 Application Development," John Wiley & Sons, Inc., Indianapolis, pp. 693–700, 2012.

[11] S. Komatineni and D. MacLean, "Pro Android 4," Apress, pp. 858–870, 2012.

[12] http://www.qrcode.com/en/ [accessed: 2014-05-08].

[13] https://github.com/zxing/zxing [accessed: 2014-05-08].

[14] M. Scott, N. Costigan, and W. Abdulwahab, "Implementing cryptographic pairings on smartcards," Proc. of the Cryptographic Hardware and Embedded Systems, pp. 134–147, 2006.



(a) (b) (c) (d)

Figure 2. (a) RkeyGenerate, (b) Role Response, (c) Role Challenge, and (d) QR Code functions of the application.