An Implementation of a Pairing-Based Anonymous Credential System with Constant Complexity

Amang Sudarsono, Toru Nakanishi, Nobuo Funabiki*

***Abstract*—An anonymous credential system allows the user to convince a verifier of the possession of a certificate issued by the issuing authority anonymously. One of the applications is the privacy-enhancing electronic ID (eID). A previously proposed anonymous credential system achieves the constant complexity for the number of user's attributes. However, due to the use of RSA-based cryptography, it still suffers from a high cost computation. Recently, we proposed an anonymous credential system with the constant complexity using a pairing-based accumulator, where more efficient ECC (Elliptic Curve Cryptography) can be used instead of the RSA. In this paper, we present an implementation of the anonymous credential system in the application to the eID. We show the practicality of the system from experimental results that the processing times are constantly less than a second.**

***Index Terms*—Authentications, privacy, anonymous credentials, electronic ID, pairings**

I. Introduction

A. Backgrounds

Electronic identification have been widely applied to access to buildings, use of facilities, Web services, etc. Currently, electronic identity (eID) such as eID card is often used for the identification. The eID is issued by a trusted organization such as the government, company, or university, and used for services provided by the organization. On the other hand, such a trusted ID is very attractive for secondary use in commercial services. The eID includes attributes of the user such as name, address, gender, occupation, date of birth. In commercial cases, instead of the identification, the attribute-based authentication is desired. For example, a service provider can refuse the access from kids, by checking the age in the eID.

One of serious issues in the existing eID systems is user's privacy. In the systems, the eID reveals the user's identity. Thus, the service provider can collect the use history of each user. Furthermore, part of attributes are a high privacy-sensitive, and only the selected attributes should be disclosed. Anonymous credential systems [1], [2], [3], [4] are one of the solutions.

Anonymous credential systems allow an issuer to issue a certificate to a user. Each certificate is a proof of membership, qualification or privilege, and it contains user's attributes. The user can anonymously convince a verifier of the possession of the certificate, where the selected attributes can be disclosed without revealing any other information about the user's privacy. The user can prove complex relations on the attributes using AND and OR relations. AND relation is used when proving the possession of all of the multiple attributes. For example, the user can prove that he is a student, he has a valid student card, and this card has not yet expired, when entering the faculty building. OR relation represents the proof for possession of one of multiple attributes. For example, he can prove that he is either a staff or a teacher when using a copy machine in the office.

* Dept. of Communication Network Engineering, Okayama University, 3-1-1 Tsushima-naka, Okayama, 700-8530 Japan
Email:nakanisi@cne.okayama-u.ac.jp

B. Related Works

In [1], Camenisch and Lysyanskaya firstly proposed an anonymous credential system based on RSA. Unfortunately, it suffers from a linear complexity in the number of user's attributes in proving AND and OR relations. Hence, this system is not suitable for small devices such as smart cards. In [3], Camenisch and Groß extended the system to solve the drawback, where the attributes are encoded into prime numbers. Then, the AND and OR relations are proven with the constant complexity w.r.t. the number of attributes using zero-knowledge proofs of integer relations on prime numbers. However, due to the use of RSA-based cryptography, it still suffers from a high cost computation.

An implementation of eID on a standard Java card using the systems of [1], [3] is shown in [5].

C. Our Contributions

In [4], we proposed a pairing-based anonymous credential system with the constant complexity. The use of pairings allows us to apply ECC (Elliptic Curve Cryptography) instead of RSA. Thus, due to the shorter key size, we expect light computations. In this paper, to show the practicality of the proposed system, we present the implementation and evaluate it from experiments. For the implementation, we suppose an eID application over Web services for mobile environments. In the prototype system, the computation time of the prover (user) and the verifier (server) are relatively constant of only 640 ms and 153 ms when proving the AND relation. They are only 630 ms and 152 ms for the OR relation. Unfortunately, as the compensation, the size of the public key depends on the number of attributes. It varies from 601 kB to 6 MB when the number of attributes increases from $1,500$ to $15,000$. In the current mobile PC environments, the data size is sufficiently practical, since the public key is not changed after it is distributed in the user registration.

II. Implemented Anonymous Credential System

A. Model

We show the model of the anonymous credential system [4] which is derived from [2]. As well as [3], this model employs two types of attributes for representation. One type is a small finite-set attribute such as the gender or the

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

occupation. Another type is a string attribute such as the full-name or the address. The example of those attributes used in the eID is depicted in TABLE I. The anonymous credential system consists of three protocols: **KeyGen**, **ObtainCert**, and **ProveCert**.

TABLE I
EXAMPLE OF STRING AND SMALL FINITE-SET ATTRIBUTES.

String	Small Finite-set	Example Values
1) full-name	8) gender	male,female
2) address	9) day of birth	1–31
3) phone number	10) month of birth	1–12
4) identity number	11) year of birth	1930–2005
5) issuance date	12) marital status	single,marriage
6) expiration date	13) nationality	193 recognized states
7) email address	14) hometown	200 allocated cities
	15) city living	200 allocated cities
	16) residence status	citizen,immigrant,...
	17) religion	moslem,christian,...
	18) blood type	A,B,O,AB
	19) occupation	student,teacher,...
	20) academic degree	B.S.,M.S,Ph.D.,...
	21) major	science,economic,...
	22) year of graduated	1970–2005
	23) workplace	200 allocated cities
	24) main language	100 allocated lang.
	25) 2nd language	100 allocated lang.
	26) topic of interest	music,sport,...
	27) favorite color	red,green,blue,...
	28) favorite music	pop,rock,jazz,...
	29) favorite sport	baseball,tennis,...

- **KeyGen**: This is a probabilistic key generation algorithm for the issuer, on the input security parameter 1^L and the maximum numbers of string and finite-set attributes n, ℓ, outputs the issuer's public key ipk and the issuer's secret key isk.
- **ObtainCert**: This is an interactive protocol between a probabilistic algorithm **User** of user with ID_i and a probabilistic algorithm **Issuer**. The common input of this protocol are ipk and (SA_i, FA_i) that are sets of string attributes and finite-set attributes of the user, respectively. In addition, the input for **Issuer** is isk. The output of **User** is the certificate $cert[i]$ ensuring the attributes (SA_i, FA_i).
- **ProveCert**: This is an interactive protocol between the user and the verifier. The common inputs are ipk, and (TSA, TFA) are subsets of string attributes and finite-set attributes respectively, and user's secret inputs are $cert[i]$ and (SA_i, FA_i).
  (1) By using the string attributes, it is proven that a part of attributes $\mathrm{TSA} \subseteq \mathrm{SA}_i$ is all certified by $cert[i]$.
  (2) For the finite-set attributes, we have two options:
    (a) the AND relation proves that all of the attribute sets TFA are certified by $cert[i]$, or
    (b) the OR relation proves that one of the attribute sets TFA is certified by $cert[i]$.

The security requirements of the anonymous credential system are as follows:

- **Unforgeability**: Only the user with the certificate certifying the proved relation is accepted by the verifier in **ProveCert** protocol.
- **Anonymity**: Any verifiers cannot identify the user from **ProveCert** protocol.
- **Unlinkability**: Any verifiers cannot determine whether any pair of **ProveCert** protocols were conducted by the same user or not.

The unlinkability is a stronger anonymity. In linkable anonymous setting, the verifier can collect the use history of an anonymous user, since the verifier can determine the sameness of the prover. Then, the history may de-anonymize the prover by relating the dates or frequency. Also, if one transaction is de-anonymized by some other method, all the transactions of the prover are de-anonymized. Therefore, the unlinkability is needed.

B. Outline of Cryptographic Construction

Generally, anonymous credential system can be constructed as follows. The certificate is a cryptographic digital signature, where multiple attribute values are signed. The signature is denoted as $Sign(a_1, a_2, \ldots, a_k)$ that is signing function on attribute values $a_1, a_2, \ldots, a_k$. By using a zero-knowledge proof technique, the user can prove only the knowledge that he owns $Sign(a_1, a_2, \ldots, a_k)$, where parts of the attributes can be secret.

In [6], using a bilinear map called pairings instead of RSA cryptography, a short certificate with the zero-knowledge proof is proposed. However, in the pairing-based certificate, each attribute value is expressed as an exponent on a base assigned to the attribute type. Hence, proving the knowledge of the certificate needs the cost depending on the number of attribute types.

In the implemented system, a pairing-based accumulator [2] is used for expressing the finite-set attributes (for the string type, it still use the exponent). The accumulator, given multiple inputs $\{g_{i_1}, \ldots, g_{i_\ell}\} \subseteq \{g_1, \ldots, g_k\}$, outputs a single value $acc = f(g_{i_1}, \ldots, g_{i_\ell})$. In addition, we can prove the validity of the accumulation with the constant cost. In our anonymous credential system, each attribute value a_i is assigned to g_i. Then, the accumulated value acc is signed by an extended signature scheme for the certificate. Thus, by the zero-knowledge proofs that g_i is accumulated to acc and that acc is signed, the user can prove that an attribute value a_i is certified with the constant complexity. In proving AND relation of $a_{i_1} \wedge \cdots \wedge a_{i_{\ell'}}$, we require the zero-knowledge proof that every g_{i_j} is accumulated into acc. The bilinear property of the pairing allows us to unify the pairing computations of the accumulators into a single pairing computation. Thus, we can achieve the constant complexity. In OR relation of $a_{i_1} \vee \cdots \vee a_{i_{\ell'}}$, we can use another accumulator to prove some a_{i_j} s.t. $a_{i_j} \in \{a_{i_1}, \cdots, a_{i_{\ell'}}\}$. The zero-knowledge proof needs only the constant cost, and thus the total cost is also constant.

The demerit of using the accumulator is that we need the number of input values of the accumulator is the number of all attribute values. Since the input values have to be included in the public key ipk, the size of ipk increases in proportion to the number of attribute values.

C. Utilized Pairing Library

To implement the pairing-based construction, we need the fast library computing the pairings together with the

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

underlying ECC operations. We utilize the library based on "Cross-twisted χ-based Ate (Xt-Xate) pairing" [7] with 254-bit group order and the embedding degree is 12. The security level is equivalent to the 3000-bit RSA. The library is based on the GMP library and implemented by C language due to the pursuit of the fastness.

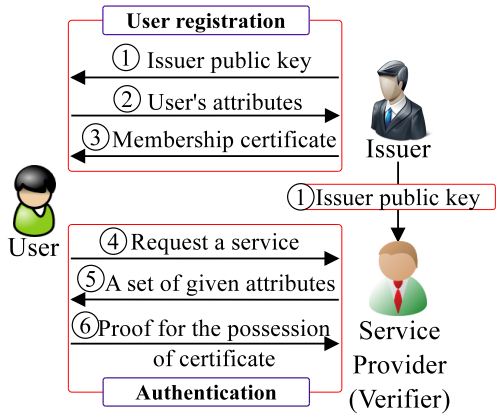

Fig. 1. System model.

III. Design of Protocols

The system model of the eID application is shown in Fig. 1. At first, the issuer publishes its public key ipk. Then, the user registers himself along with particular attributes (SA_i, FA_i) to the issuer for certification by using **ObtainCert** protocol via a secure channel (①∼③). Based on the issued certificate, he requests a service to the Service Provider (SP) (④). Then, the SP specifies attributes that the SP wants to know (⑤). This specification forms AND or OR relation, depending on the SP's requirement. Then, the user generates a proof for the possession of certificate w.r.t. the specified attribute(s) and shows it to the SP (verifier) anonymously by using **ProveCert** protocol (⑥). If only if the verification of user's proof is valid, the SP grants the user to access a requested service. In this scenario, we can consider that the number of finite-set attribute values is at least $1,500$, due to the involvement of the nationality, city, language, and other categories as shown in TABLE I.

We design our protocols: user registration and authentication as follows.

A. User Registration

Fig. 2 shows the user registration protocol. This protocol comprises the following two steps. In advance, the user has to fetch the issuer public key ipk, for example by downloading it from the official web-site of issuer. We assume that there is an existing access control application (e.g., based on username and password) to permit the user to use a communication channel (which is out of scope of this paper). Then, he can use such channel to perform **ObtainCert** protocol together with the issuer.

(1) Registration of user's attributes.
The user requests the registration. He sends his attributes for certification (①∼③).

(2) Issuing the certificate.

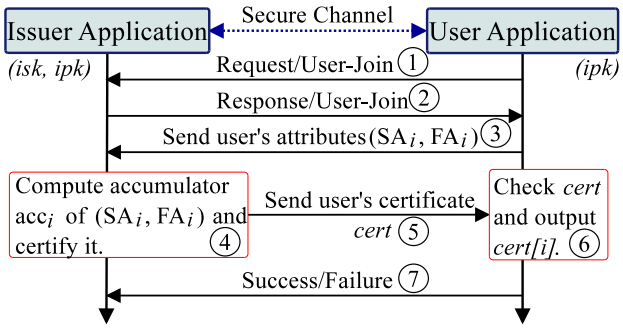

Fig. 2. User registration protocol.

(a) The issuer computes the accumulator, certifies it into certificate $cert$ and sends $cert$ to the user (③∼⑤).

(b) The user checks the validity of $cert$ to ensure whether $cert$ was sent by the legitimate issuer or not. If it is valid, he embeds his secret keys into the $cert$ and outputs the user certificate $cert[i]$ (⑥∼⑦).

B. Authentication

The authentication mechanism is performed by using a **ProveCert** protocol. Fig. 3 shows the authentication process to allow the user to access the service provided by an SP through wireless networks. This protocol comprises the following two steps. In this protocol, the user can prove his possession of particular attributes into two ways, AND relation proving and OR relation proving.

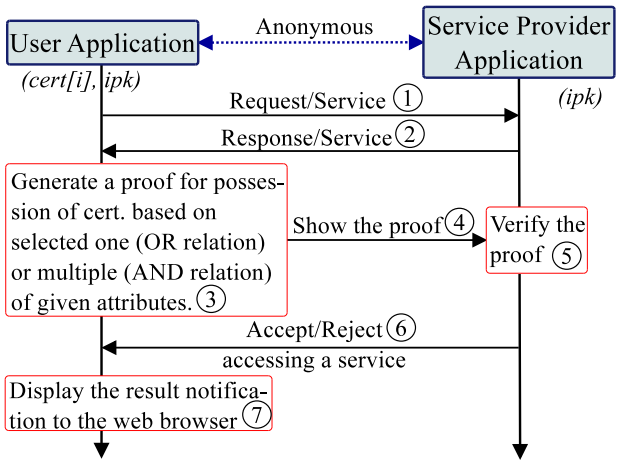

Fig. 3. Authentication protocol.

(1) Generation and transmission of a proof for possession of certificate.

(a) The user requests a service to the SP. Then, the SP provides him a set of specified attributes. Depending on the SP's requirement, there are two options, multiple selection and one selection of a set of specified attributes (①∼②).

(b) By using $cert[i]$ and the selected attribute(s), the user generates the proof for the possession of such attribute(s). This proof means that

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

such selected attribute(s) is(are) included in the certificate. Then, he shows the proof to the SP (③∼④).

(2) Verification of a proof for possession of certificate.

(a) The SP verifies the proof. If it is valid, the SP grants the user to access the service (resp. reject the service) (⑤∼⑥).

(b) The verification result is displayed in the web browser of the user which indicates either accept or reject to access the service (⑦).

IV. Implementation

We basically implemented our system using Java through the Java GUI and Java applet at the user and Java servlet at the Servide Provider (SP) and the issuer, since the Java applet and Java servlet communications are useful to implement the web-based applications. The communication between the user and the servers (i.e., the issuer and SP) is over http connections. The cryptographic processes in the anonymous credential system are implemented by C language as the middle-ware, since we need the fast computations as well as the underlying pairing library. As the interface between C and Java, we use Java Native Interface (JNI).

Pre-computation process. To reduce the computation time in the authentication protocol, we pre-compute the fixed value components (e.g., the pairing calculations of fixed values of public key components). This pre-computed data are used to generate or verify the proof, where the computation times of generating and verifying the proof are reduced to about 51% and 30%, respectively.

Reply attack protection. An attacker may duplicate some valid transcript of the authentication to use for another session. To protect this, we use a random nonce. The nonce is computed by the hash function from the user's defined message (this can be any messages including random numbers) and timestamps.

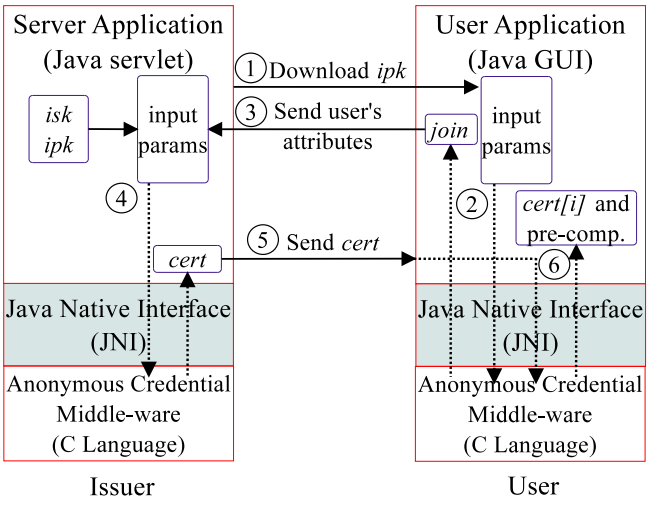

Fig. 4. Implementation of user registration protocol.

The implementation of user registration protocol is depicted from Fig. 4. In this implementation, there are three steps:

1) Step 1: By using the ipk, he executes the registration process of the middle-ware through JNI. Then, he sends the issuer his attributes (①∼③).
2) Step 2: User's attributes, ipk, and issuer secret key isk are the input parameters for generating user membership certificate $cert$ provided by the middle-ware (④∼⑤).
3) Step 3: Upon receiving $cert$, the user embeds his secret to the $cert$ and outputs the user certificate $cert[i]$. Then, he executes a pre-computation process to provide a pre-computed data to be used later in generating the proof (⑥).

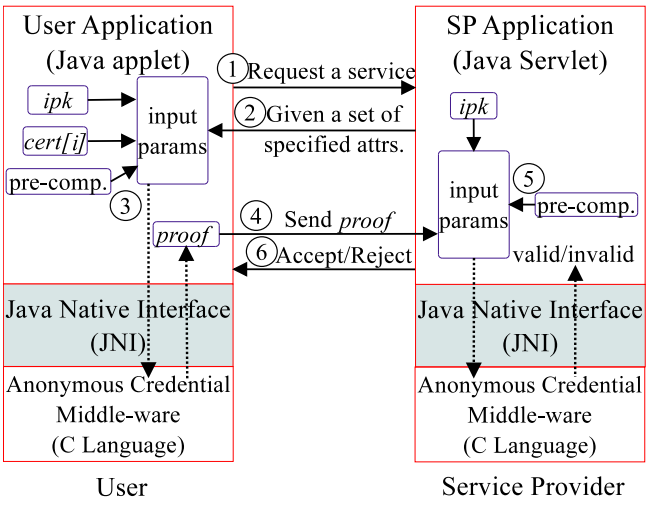

Fig. 5. Implementation of authentication protocol.

The implementation of authentication protocol is depicted from Fig. 5. In this implementation, there are two main processes; user's proof generation and verification. At first, the SP needs to download the ipk from the issuer and executes the pre-computation process from the middle-ware to obtain pre-computed data.

1) User's proof generation: The selected attribute(s), ipk, $cert[i]$, and user's pre-computed data are used as the input parameters for generating the proof. This is performed by using the proving function in the middle-ware. The function provides AND and OR relations proving. The proof is based on the random nonce generated from the hash function. The Java applet is embedded into local web page to interact with Java servlet at the SP's web server (①∼④).
2) Verification: At the SP side, the received proof, ipk, and the SP's pre-computed data are used as the input parameters to verify the proof. This is done by executing the verification function in the middle-ware. This function also provides AND and OR relations. The correctness of the nonce is also checked. The output of this function is valid or invalid (resp. accept or reject) (⑤∼⑥).

V. Experimental Results

A. Test Bed

We have implemented the user registration protocol and authentication protocol (Fig. 2 and Fig. 3) in our proposed

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

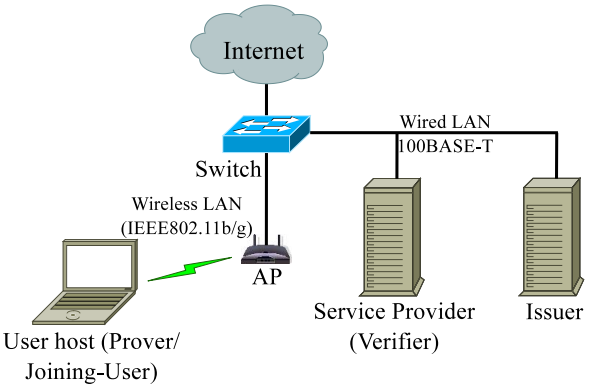

Fig. 6. Experimental environment.

system to confirm the practicality. Fig. 6 shows the experimental environment of our proposed system. The devices and software specifications used in this test bed are shown in TABLE II.

TABLE II
SPECIFICATION OF DEVICES IN EXPERIMENTS.

Verifier/ Issuing Authority	Intel Core2Duo 3GHz, 4GB RAM Ubuntu Linux 9.10 kernel-2.6.31 gcc-4.4.1, OpenSSL-0.9.8g, GMP-4.3.1 Java-1.6.0_18, apache-tomcat-5.5.28
Prover/ Joining-User	Intel Atom 1.33GHz, 1GB RAM Windows XP Home Edition SP3 Atheros AR928X Wireless Adapter MinGW-5.1.6, OpenSSL-0.9.8g, GMP-4.3.1, Java-1.6.0_19

B. Evaluation

Fig. 7 shows the total processing times of the authentication protocol in the AND relation and the user registration protocol. The total time of the authentication protocol includes the proving time, verification time, and the communication time during the process. This time varies from 917 ms to 927 ms, where the number of attributes are from $1,500$ to $15,000$. It is from 904 ms to 912 ms for the OR relation. The communication time is measured from sending the proof until receiving the response of authentication process from the verifier with excluding the verification time. The communication time is about 120 ms. The time of the user registration protocol varies from 809 ms to 828 ms, as the number of attributes increases from $1,500$ to $15,000$. This small variation is derived from accumulator computation and fetching the public key. In addition, our system has an advantage that the sizes of the proofs for AND and OR relation proving are constant at about 1324 Byte and 1424 Byte, respectively. The size of issued certificate is constant at about 272 Byte.

Fig. 8 shows the processing times of the proving and verification for the AND relation and the size of the public key as the number of attributes varies. The times of the proving and verification vary from 642 ms to 649 ms and from 153 ms to 154 ms, respectively. This small variation is derived from the process of fetching the public key. However, these times are constant by ignoring such process. On the other hand, the times in proving the OR relation are 635 ms and 152 ms. Unfortunately, this system has a drawback that

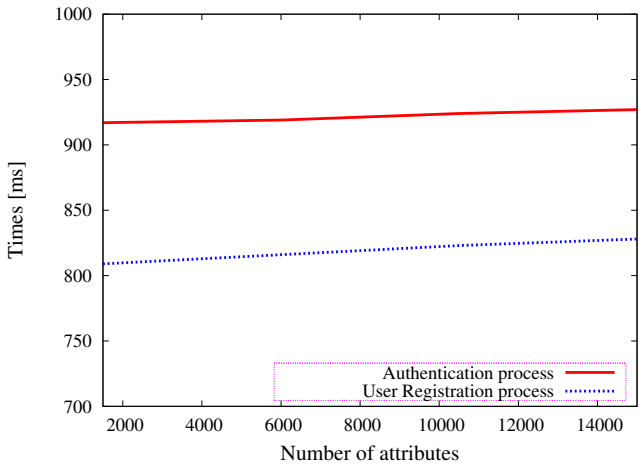

Fig. 7. Total processing times.

the size of public key varies from 601 kB to 6 MB. However, in the current mobile PC environments, the data size is sufficiently practical, since the public key is not changed after it is distributed in the user registration.

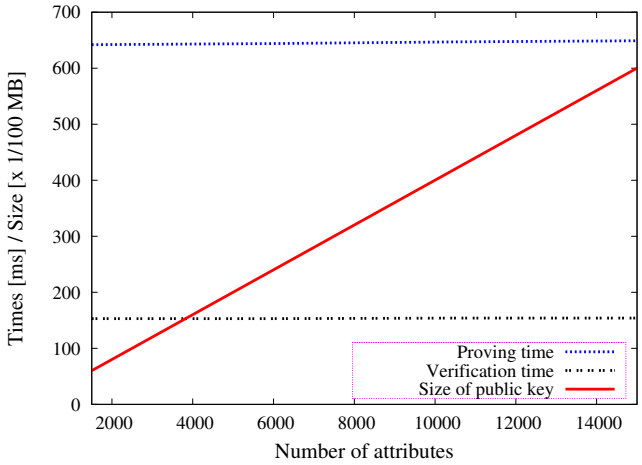

Fig. 8. Processing times and size of public key.

VI. CONCLUSION

We have presented an implementation of a pairing-based accumulator for the anonymous credential system with efficient proofs of AND and OR relations of attributes to the application of eID. In the prototype system, the user registration and authentication protocols are processed within a second, and thus we conclude that the system is sufficient practical. To achieve more efficient proofs and reduce the size of the public key are our future works.

ACKNOWLEDGMENT

This work was partially supported by Grant-in-Aid for Scientific Research (21300004) from Japan Society for the Promotion of Science (JSPS).

REFERENCES

[1] J. Camenisch and A. Lysyanskaya, “Dynamic accumulators and application to efficient revocation of anonymous credentials,” in *Advances in Cryptology — CRYPTO 2002*, ser. LNCS 2442. Springer–Verlag, 2002, pp. 61–76.

[2] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials," in *Proc. 12th International Conference on Practice and Theory in Public Key Cryptography (PKC 2009)*, ser. LNCS 5443. Springer–Verlag, 2009, pp. 481–500.
[3] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," in *Proc. ACM Conference on Computer and Communications Security 2008 (ACM-CCS'08)*, 2008, pp. 345–356.
[4] A. Sudarsono, T. Nakanishi, and N. Funabiki, "Efficient proofs of attributes in anonymous credential systems using a pairing-based accumulator," in *Computer Security Symposium 2010 (CSS2010)*, 2010, pp. 801–806.
[5] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup, "Anonymous credentials on a standard java card," in *Proc. ACM Conference on Computer and Communications Security 2009 (ACM-CCS'09)*, 2009, pp. 600–610.
[6] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology — CRYPTO 2004*, ser. LNCS 3152. Springer–Verlag, 2004, pp. 41–55.
[7] M. Akane, Y. Nogami, and Y. Morikawa, "Fast Ate pairing computation of embedding degree 12 using subfield-twisted eliptic curve," *IEICE Trans. Fundamentals*, vol. E92-A, no. 2, pp. 508–516, 2009.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)