Efficient Access Control of Sensitive Data Service in Outsourcing Scenarios

Yang ZHANG and Jun-Liang CHEN

State Key laboratory of Networking and Switching Technology,
Beijing University of Posts & Telecommunications,
Beijing 100876, China
YangZhang@bupt.edu.cn

Abstract—With the rapid application of service-oriented technologies, service and data outsourcing has become a practical and useful computing paradigm. Combined use of access control and cryptography was proposed by many researchers to protect information in this outsourcing scenario. However, existing approaches often limit dynamical update of access control policy, or have security weakness in practical use. In this paper, we propose a new solution to realize efficient access control of sensitive data service in outsourcing scenarios by using a new re-encryption execution model. Our solution realizes selective access control, dynamical policy updating, simple key management, and collusion prevention of the outsourcee and customers. We also give some proofs of our implementation.

Keywords—Data Outsourcing, Re-encryption Scheme, Access Control, Identity Metasystem.

I. INTRODUCTION

With the rapid growth of service and data outsourcing such as in cloud computing environments, customers often delegate the service of storing and managing their sensitive data to third parties. In this scenario, the assumption that access to resources is controlled by an omniscient reference monitor executing perfect surveillance on requests is becoming more and more impractical [1]. Integrating encryption policies on resources with the access regulations is a promising approach to protect customers' content.

As an integration example, the selective encryption technique was adopted to encrypt different data with different keys, which guarantees that legal users can retrieve the key to decrypt the encrypted resource [2], [3], [4], [5], [1]. However, those approaches need user to derive all the keys of authorized data items according to a public catalog of tokens, which results in static access control policy and hard to update. Therefore, A DSP re-encryption mechanism was proposed by [6], which can implement the selective access control of the encrypted data, relieve the users of the client from the complex key derivation procedure, and provide dynamic policy updating. Unfortunately, if the server colludes with a user, the collusion user may access to all the unauthorized data items as long as the server re-encrypt each data item by using the re-encryption key of the collusion user.

In this paper, we propose a new access control solution to solve the above problems. Being similar to the work of [6], we also adopt the concept of proxy re-encryption [7], [8] to implement selective access control, dynamical policy updating, and simple key management. Besides that, we introduce privilege-value to prevent colluded decryption of unauthorized data items by using a new re-encryption model in outsourcing environments (BRM) which is a kind of execution method of proxy re-encryption scheme. Contributions of our paper are as follows:

1. A new architecture is presented to realize efficient access control of sensitive data in outsourcing scenarios, which clarifies key components to achieve the goal as well as easy to use.
2. A new re-encryption model in outsourcing environments (BRM) is presented as well as its implementation.
3. Based on the architecture and BRM, the solution of efficient access control enforcement is proposed.

The rest of this paper is organized as follows. Section 2 gives a description on preliminaries. In section 3, our solution of access control enforcement is described. Section 4 focuses on the implementation of BRM. In section 5, proofs are given. Finally, conclusions are drawn in section 6.

II. PRELIMINARIES

A. Key-private Proxy Re-encryption Scheme Π

Suppose we have groups G and G_T of the same prime order q and security parameter κ. Assume the discrete logarithm problem is hard in both groups. Then we need a cryptographic bilinear map $e: G \times G \rightarrow G_T$ to satisfy the following properties [9], [10]:

1) Bilinearity: $\forall a,b \in Z_q$, $P,Q \in G$, $e(aP,bQ) = e(P,Q)^{ab}$.

2) No-degeneracy: For any point $P \in G$, $e(P,P) \neq 1_{G_T}$.

3) Computability: there exists an efficient algorithm to compute $e(P,Q)$ for $\forall P,Q \in G$.

The key-private proxy re-encryption scheme $\Pi = (Setup, KeyGen, RekeyGen, Enc, ReEnc, Dec)$ is as follows:

$Setup$: Generate (q,g,G,G_T,e), where $<g>=G$. Choose a random generator $h \in G$. Compute $Z = e(g,h)$, and set the public parameter $PP = (g,h,Z)$.

KeyGen : Choose random values $a_1, a_2 \in Z_q$ and set the public key as $pk = (Z^{a_1}, g^{a_2})$ with secret key $sk = (a_1, a_2)$.

ReKeyGen :A user A with secret key $sk = (a_1, a_2)$ can delegate to a user B with public key (Z^{b_1}, g^{b_2}) as:

1. Select random values $r, w \in Z_q$.
2. Compute $rk_{A \to B} = (R_1, R_2, R_3, R_4) = (g^{b_2(a_1+r)}, h^r, Z^{b_2 w}, Z^w)$

Enc : To encrypt a message $m \in G_T$ under public key $pk_A = (Z^{a_1}, g^{a_2})$, do:

1. Select a random value $k \in Z_q$.
2. Compute the ciphertext $C = (\mu, \nu, \gamma) = (g^k, h^k, mZ^{a_1 k})$.

ReEnc : Given a re-encryption key $rk_{A \to B} = (R_1, R_2, R_3, R_4)$ and $C = (\mu, \nu, \gamma)$, do:

1. Verify the ciphertext is well-formed by checking : $e(\mu, h) = e(g, \nu)$. If this does not hold, output $\perp$ and abort.
2. Otherwise, compute $t_1 = e(R_1, \nu) = Z^{b_2 k(a_1+r)}$, $t_2 = \gamma e(\mu, R_2) = mZ^{k(a_1+r)}$.
3. Select a random value w' .
4. Compute $\alpha = t_1 R_3^{w'} = Z^{b_2(k(a_1+r)+ww'}$, $\beta = t_2 R_4^{w'} = mZ^{k(a_1+r)+ww'}$.
5. Output $C_B = (\alpha, \beta)$.

Dec : Given secret key (b_1, b_2) , decrypt $C_B = (\alpha, \beta)$ as follows:

$m = \beta / \alpha^{1/b_2}$.

B. Enhanced Identity Metasystem

The goal of an identity metasystem is to guarantee its users a simple, consistent experience. Its model has three roles: the **resource**, the **identity provider** (IdP) and the **identity selector system**. A resource is any system capable of authenticating users based on their information cards. The IdP issues information cards to user and provides authentication service. The identity selector system is a core component to handle messages between a resource and an IdP. A user can use the identity selector system to securely manage his information cards.

Fig. 1 illustrates the basic architecture of identity metasystem. The identity selector system is main part of the metasystem model. The underlying is backup services such as local IdP, Secure Storage, and Card Sync. Such identity metasystems include Microsoft Cardspace [11], [12] and Novel Bandit project [13].

A personal identity metasystem often runs on her trusted device. The device can operate as a self-issued identity provider. The identity selector can run on an untrusted terminal. For the ever-increasing service and data outsourcing, identity metasystems are enhanced with access control service such that the outsourcer can decide who has the privilege of access to her sensitive data stored by the outsourcee.

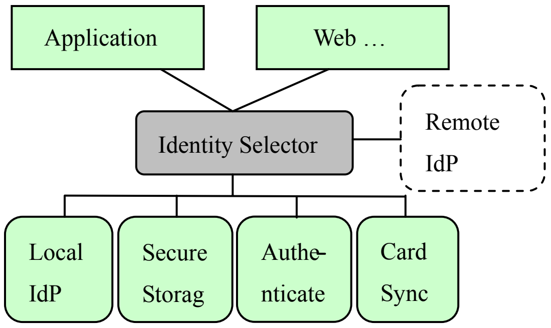

Figure 1. Architecture of Identity Metasystem

III. Access Control Solution

In this section, we first describe the basic re-encryption model (BRM) which specifies how to use re-encryption scheme Π in outsourcing environments. Then, based on BRM and enhanced identity metasystems, the architecture of our solution is presented. Finally, the solution of efficient access control enforcement is given to realize the control of sensitive data dissemination.

A. Basic Re-encryption Model in Outsourcing Environments

The basic re-encryption model (BRM) describes how to use re-encryption scheme Π in outsourcing environments, where encryption keys are integrated with access control policies. If the customer B is legal user, the outsourcee P sends her the re-encrypted data, and the owner A computes for her some privilege-value which can be used in the decryption algorithm with the secret key of B. BRM consists of five algorithms as follows:

1. *Encrypt* : The owner A encrypts sensitive data and sends to the outsourcing P .
2. *RekeyGen* : When the outsourcee P need re-encrypt the selected sensitive data from her storage for a customer B, the owner A computes the re-encryption key and sends it to P .
3. *ReEncrypt* : P re-encrypts the sensitive data selected from her storage according to the request of the customer B and sends the re-ciphertext to B .

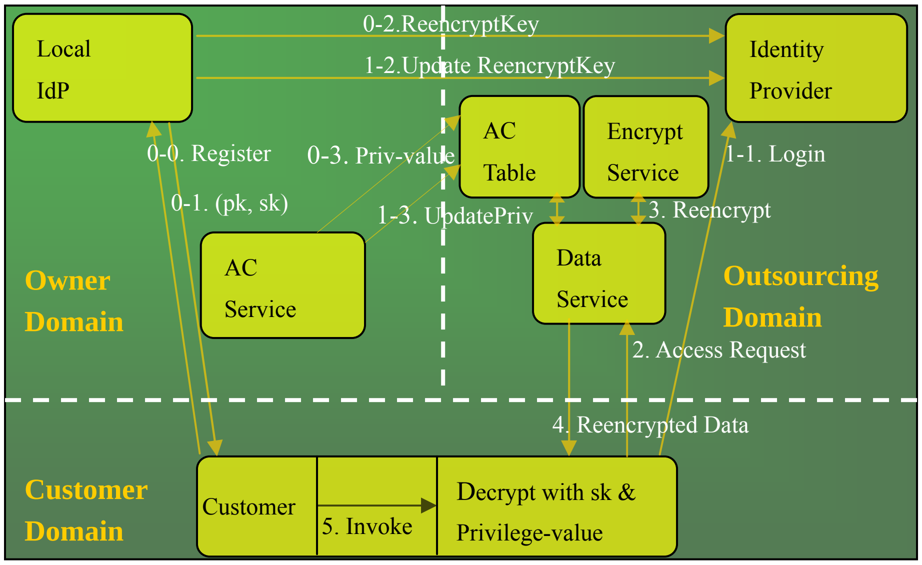

Figure 2. The Architecture of ReAC

4. *CalculatePrivilege* : The owner *A* calculates the privilege-value according to the requested data and the identity of *B*, and sends it to *B*.
5. *Decrypt* : *B* decrypts the re-ciphertext to get the sensitive data with her secret key and the privilege-value.

B. The Architecture of Our Solution

In order to efficiently manage access control of sensitive data in outsourcing scenarios, we propose a re-encryption-based access control solution (ReAC). Participants in this solution include a data owner *A* (a natural person or a company), a data outsourcee *P,* and a customer *B. A* encrypts her sensitive data and sends it to *P*. *P* stores and manage these encrypted data. *B* accesses some parts of these data from *P.* Each participant lies in her own security domain such as owner domain, outsourcing domain and customer domain. For each end user, enhanced identity metasystem is equipped, which supports easy-to-use, consistent experience, security transparency, and access control in outsourcing scenarios.

Figure 2 illustrates our solution architecture which includes three security domains: an owner domain, an outsourcing domain and a customer domain. Each security domain has key entities with respect to our access control solution as follows.

1. There exits enhanced identity metasystem in the owner domain and customer domain.

2. **Owner Domain**

- AC service. The access control service (AC service) in the owner domain is charge of managing access to the outsourced sensitive data, i.e. it inserts, updates, deletes items in the access control policy table (AC Table) of the outsourcing domain.
- Local IdP. The local identity provider service (IdP) issues/exchanges security tokens and makes identity claims in issued security tokens.

3. **Outsourcing Domain**

- Data Service. The data service provides interface to read and write encrypted sensitive data. When the customer *B* makes a request, the service verifies the privilege of *B* according to the AC Table and invokes the encrypt-service to re-encrypt the encrypted data.
- Encrypt-service. The encrypt-service re-encrypts the encrypted data with the re-encryption key of *B.*
- AC Table. The access control policy table (AC Table) stores user’s access matrix as table 1 where ind1 and ind2 denote as the indicators of sensitive data item, B1 and B2 denote as customers, PV denotes as privilege-value, and digital “0” means the customer has not right to access this data item.

Table 1. Access Matrix

	Ind1	Ind2	…
B1	PV	0	
B2	0	PV	
…			

- Identity Provider. As Local IdP.

The followings are some description of Fig. 2 in order:

0-0. The customer *B* registers herself to the Local IdP in the owner domain.

0-1. The Local IdP returns a public/private key pair (pk_B, sk_B) to *B* while the owner *A* has another public/private key pair (pk_A, sk_A).

0-2. The Local IdP computes a re-encryption key for *B* according to (pk_B, sk_B) and (pk_A, sk_A), and sends the re-encryption key and *B's* public key to the outsourcee *P*.

0-3. The AC service computes the privilege values (Priv-value) for *B* and invokes the AC Table service to insert an AC (Access Control) record of *B* into the AC Table of *P*.

1-1. Before it can access the sensitive data of *A* outsourced in *P*, *B* single signs on the Identity Provider of *P*. If the authentication is successful, the identity provider returns access security tokens to *B*.

1-2. After the login of *B*, the Local IdP may refresh the re-encryption key for *B*.

1-3. After the login of *B*, the Access Service may refresh the privilege of *B* because of the change of re-encryption key or modification instructions. It modifies the AC Table of *P* to update privilege values for *B*.

2. *B* makes request to access some sensitive data using the access tokens.

3. The Data Service checks the AC Table. If *B* has the privilege to access the sensitive data, the service invokes the encrypt-service to re-encrypt the encrypted data.

4. The Data Service returns the re-encrypted data and the corresponding privilege-value of *B* to *B*.

5. *B* decrypts the re-encrypted data to get the sensitive data with her secret key and the privilege-value.

C. Efficient Access Control Enforcement

According to the architecture, efficient access control can be achieved in outsourcing scenarios as follows, where re-encryption key table is stored as in table 2.

Table 2. Re-encryption key Table

	Rekey
B1	Key1
B2	Key2
…	

Preparative phase 1: The data owner *A* encrypts her sensitive data. She can choose two policies to encrypt it. That is to say, the data is directly encrypted with the asymmetric key pk_A of *A*, or the items of the data are encrypted with symmetric keys and the symmetric keys are encrypted with the asymmetric key pk_A of *A*. The encrypted data and the possible encrypted symmetric keys are stored by the outsourcee *P*.

Preparative phase 2: The Local IdP of *A* issues a public/private key pair (pk_B, sk_B) for each customer *B*. At the same time, the $RekeyGen$ algorithm in BRM is used to generate a re-encryption key which is stored by *P* as in Table 2.

Policy Processing: Although the access control policy table is stored by *P*, the policy is constructed by *A*. For a customer *B*, the AC service of *A* uses the $CalculatePrivilege$ algorithm in the BRM to compute *B*'s her privilege-value and update the AC Table with the privilege-value.

Verification phase: Verify whether the requesting B has the privilege to access the data items she requests. If she does, P continues the process, or else terminates the process.

Retrieving phase: Encryption-service retrieves the re-encryption key of *B* from the re-encryption key table.

Re-encryption phase: Encrypt-service adopts the $ReEncrypt$ algorithm in BRM to re-encrypt the encrypted data with the retrieved re-encryption key. According to the type of encryption of data (asymmetric or symmetric), an appropriate re-encryption procedure is adopted which is described in Section 4. The re-encrypted data is returned to *A* who uses the $Decrypt$ algorithm in BRM to get the sensitive data.

IV. Implementation of Basic Re-encryption Model

A. Using Asymmetric Encryption on Sensitive Data

When the sensitive data is directly encrypted with the asymmetric public key of the owner *B*, the BRM is implemented as follows.

1. $Encrypt$: For the sensitive data m , the owner *A* encrypts it using the asymmetric encryption algorithm in the re-encryption scheme Π , i.e. *A* selects a random value k to encrypts m with her public key : $C_s = Enc_{pk_A}(m)$. *A* sends (Ind_m, C_s) to the outsourcee *P* who stores and manages (Ind_m, C_s) for *A*, where Ind_m indicates m such as a index of m or a keyword on m . *A* herself stores (Ind_m, k) .
2. $RekeyGen$: After invoking $ReKeyGen$ algorithm in Π , *A* gets $ReKey = (B,(r,w),(R_1,R_2,R_3,R_4)))$ $= (B,(r,w),(g^{b_2(a_1+r)},h^r,Z^{b_2 w} = e(g,h)^{b_2 w},Z^w = e(g,h)^w)$ for customer *B* and computes $ReKey' = (R_1,R_2',R_3,R_4) = (R_1,h,R_3,R_4)$. *A* sends $ReKey'$ to *P* as the re-encryption key of *B* and stores $(B,(r,w))$ herself.
3. $ReEncrypt$: As in Π , *P* computes $C_R = (\alpha,\beta) = ReEnc(C_s)$ sends C_R to B .
4. $CalculatePrivilege$: Based on the Ind_m of the requested data and the identity of B , A gets (Ind_m, k) and $(B,(r,w))$. Then, *A* calculates $Privilege - value = h^{kr-k}$ and sends it *P*.
5. $Decrypt$: Given C_R , *B* decrypts the ciphertex $C_R = (\alpha,\beta)$ with her secret key (b_1,b_2) and the privilege-value: $m = e(g, Privilege - value)\beta / \alpha^{b_2}$.

The correction of this implementation can be illustrated as follows.

$Encrypt$: $C_s = Enc_{pk_A}(m) = (\mu,\nu,\gamma) = (g^k, h^k, mZ^{a_1 k})$

$ReEncrypt$: $C_R = (\alpha,\beta) = ReEnc(C_s)$

$$(R_1,R_2,R_3,R_4) = (g^{b_2(a_1+r)}, h, Z^{b_2 w}, Z^w)$$

$$t_1 = e(R_1,\nu) = Z^{b_2 k(a_1+r)} , t_2 = \gamma e(\mu,R_2) = mZ^{ka_1+k} .$$

$$\alpha = t_1 R_3^{w'} = Z^{b_2(k(a_1+r)+ww'}, \quad \beta = t_2 R_4^{w'} = mZ^{ka_1+k+ww'} .$$

$Decrypt$:
$e(g, Privilege - value)\beta / \alpha^{b_2} = e(g, h^{kr-k})\beta / \alpha^{b_2} = Z^{kr-k} mZ^{k-kr} = m$

B. Using Symmetric Encryption on Sensitive Data

When the sensitive data is encrypted with symmetric keys selected by *B*, the BRM is implemented as follows.

1. $Encrypt$: For the sensitive data m , the owner *A* first encrypts it using symmetric encryption algorithm with a symmetric key K_{sym} : $C_s = SymE_{K_{sym}}(m)$. Then *A* encrypts K_{sym} using the asymmetric encryption algorithm in the re-encryption scheme Π , i.e. *A* selects a random value k to encrypt K_{sym} with her secret key : $C_k = Enc_{pk_A}(K_{sym})$. A stores (Ind_m, k, K_{sym}) where Ind_m indicates m such as a index of m or a keyword on m . A sends (Ind_m, C_s, C_k) to the outsourcee *P* who stores and manages (Ind_m, C_s, C_k) for *A*.
2. $RekeyGen$: After invoking $ReKeyGen$ algorithm in Π , *A* gets $ReKey = (B,(r,w),(R_1,R_2,R_3,R_4)))$ $= (B,(r,w),(g^{b_2(a_1+r)},h^r,Z^{b_2 w} = e(g,h)^{b_2 w},Z^w = e(g,h)^w)$ for customer *B* and computes $ReKey' = (R_1,R_2',R_3,R_4) = (R_1,h,R_3,R_4)$. *A* sends $ReKey'$ to *P* as the re-encryption key of *B* and stores $(B,(r,w))$ herself.
3. $ReEncrypt$: As in Π , *P* computes $C_1 = (\alpha,\beta) = ReEnc(C_k)$, $C_1' = Enc_{pk_B}(K_{sym}')$ where K_{sym}' is symmetric key selected by *P*. Then, *P* computes $C_2 = SymE_{K_{sym}'}(C_s)$ and sends (C_1, C_1', C_2) to B .
4. $CalculatePrivilege$: Based on the Ind_m of the requested data and the identity of B , A gets (Ind_m, k, K_{sym}) and $(B,(r,w))$. Then, *A* calculates $Privilege - value = h^{kr-k}$ and sends it *P*.
5. $Decrypt$: Given (C_1, C_1', C_2) *B* decrypts the ciphertext $C_1 = (\alpha,\beta)$ with his secret key (b_1,b_2)

and privilege-value $K_{sym} = e(g, Privilege-value)\beta/\alpha^{b_2}$. Then, B decrypts C_1' using normal asymmetric decryption algorithm with her secret key : $K_{sym}' = Dec_{sk_B}(C_1')$. Finally, B decrypts C_2 with K_{sym} and K_{sym}' : $m = SymD_{K_{sym}}(SymD_{K_{sym}'}(C_2))$.

V. Proofs

For a unique data item, there are multiple privilege-values for multiple customers. The privilege-value h^{kr-k} includes secret random value k used in asymmetric encryption algorithm of Π. If k can be computed according to the multiple privilege-values, the sensitive data m in $C = (\mu, \nu, \gamma) = (g^k, h^k, mZ^{a_1k})$ can be disclosed according to A's public key $pk = (Z^{a_1}, g^{a_2})$: $m = \gamma/(Z^{a_1})^k$. **Theorem 1** specifies this event will not take place.

Theorem 1. Given h^{kr_1-k} and h^{kr_2-k} where $h \in G_T$ and $k, r_1, r_2 \in Z_p$ are as in the re-encryption scheme Π, it is hard to compute k.

Proof. Assume that there exits an algorithm *adv* to compute k. We uses *adv* as a sub-algorithm to solve the discrete log problem $y = h^x$. Select two random values r_1, r_2, compute $y_1 = (y)^{r_1}/y = h^{xr_1-x}$, $y_2 = (y)^{r_2}/y = h^{xr_2-x}$. Invoke *adv* with y_1, y_2 as input, and get the output x of *adv.* That is to say, we solve the discrete log problem $y = h^x$. Therefore, it is hard to compute k given h^{kr_1-k} and h^{kr_2-k} as in Π.

Therefore, privilege-values are not only used to dynamically update access control policy of the sensitive data, but also used to prevent the collusion of P and customers from disclosing those unauthorized sensitive data.

VI. Conclusions

In this paper, we address the problem of how to efficiently enforce access control in service and data outsourcing scenarios to protect the sensitive data of owners. The new re-encryption execution model is first presented as a foundation of our solution. Then, the architecture of the solution is described. Based on the model and architecture, the solution is designed to efficiently enforce access control. Our solution realizes selective access control, dynamical policy updating, simple key management, and collusion prevention of the outsourcee and customers. Therefore, our solution will provide strong building blocks for the design and implementation of protecting information in outsourcing scenarios such as cloud computing.

Reference

[1] S. De Capitani di Vimercati, S. Foresti, S. Jajodia. Preserving Confidentiality of Security Policies in Data Outsourcing. Proceedings of the 7th ACM workshop on Privacy in the electronic society, pp. 75-84, 2008.

[2] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati. Selective Data Encryption in Outsourced Dynamic Environments. Electronic Notes in Theoretical Computer Science, pp. 127-142, 2007.

[3] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati. Metadata Management in Outsourced encrypted databases. Lecture Notes in Computer Science, Secure Data Management, pp.16-32, 2007.

[4] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati. Over-encryption: management of access control evolution on outsourced data. Proc. of the 33rd VLDB Conference, Vienna, Austria, September, 2007, pp.123-134, 2007.

[5] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati. A data outsourcing architecture combining cryptography and access control. Proc. of the 1st Computer Security Architecture Workshop, Fairfax, VA, November 2007, pp. 63-69.

[6] X. Tian, X. Wang, A. Zhou. DSP RE-Encryption: A Flexible Mechanism for Access Control Enforcement Management in DaaS. 2009 IEEE International Conference on Cloud Computing, pp. 25-32, 2009.

[7] G. Ateniese, K. Benson, and S.Hohenberger. Key-Private Proxy Re-encryption. Topics in Cryptology - CTRSA, LNCS,vol.5473, pp. 279-294, 2009.

[8] B. Libert and D. Vergnaud. Tracing Malicious Proxies in Proxy Re-encryption. Pairing-Based Cryptography - Pairing, LNCS,vol.5209, pp.332-353, 2008.

[9] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In Proc. of CRYPTO'01, volume 2139, pp. 213-229, 2001.

[10] P. Barreto, H. Kim, B. Bynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In Proc. CRYPTO'02, pp. 354-368, 2002.

[11] D. Chappell. Introducing Windows CardSpace. Windows Vista Technical Articles, April 2006 http://msdn2.microsoft.com/en-us/library/aa480189.aspx

[12] CodeIdol.com. InfoCard Architecture and Security http://codeidol.com/csharp/indigo/InfoCard/InfoCard-Architecture-and-Security/

[13] Novell corp. Bandit project. http://www.bandit-project.org/index.php/Welcome_to_Bandit