CloudSeal: End-to-End Content Protection in Cloud-based Storage and Delivery Services

Huijun Xiong§, Xinwen Zhang†, Wei Zhu†, and Danfeng Yao§

§Computer Science Department, Virginia Tech, Blacksburg, VA, USA
{huijun,danfeng}@cs.vt.edu
†Huawei Research Center, Santa Clara, CA, USA
{xinwen.zhang,wei.zhu}@huawei.com

Abstract. Recent years have seen the trend to leverage cloud-based services for large scale content storage, processing, and distribution. Security and privacy are among top concerns for public cloud environments. Towards the end-to-end content confidentiality protection, we propose *CloudSeal*, a scheme for securely sharing and distributing data via cloud-based data storage and content delivery services (e.g., Amazon S3 and CloudFront). CloudSeal ensures the confidentiality of content stored in public cloud storage services, by encrypting it before sharing at the cloud. To achieve flexible access control policies, CloudSeal further adopts k-out-of-n secret sharing and broadcast revocation mechanisms to renew shared secrets, e.g., when a user joins or leaves a content sharing group. Most importantly, CloudSeal leverages proxy re-encryption algorithm to transfer part of stored cipher content in the cloud, which can be decrypted by a valid user with updated secret keys. We achieve this property without modifying most of the encrypted content. This feature is critical for the efficiency of content distribution.

1 Introduction

Security issues have been one of the top concerns for cloud computing [1], despite the increase in cloud usage. Among them, how to maintain the confidentiality and privacy of outsourced content in the public cloud remains a challenging task. The issue becomes more difficult with flexible content processing and sharing among Internet users through cloud-based services. For confidentiality, a content provider should encrypt her content with keys that are out of the reach of the cloud provider. Content accessible to different users should be encrypted with different keys to distinguish their privileges. Key management may be complex when the content is shared by many users with different privileges. Previous work has studied such problems in conventional distributed environments [2, 3]. For large scale cloud-based content sharing and distribution services, there are additional new requirements besides key management as explained in the following. First, the accessible content of a user may change dynamically, e.g., based on the content provider's security policy or the user's subscription information. Each piece of content may be shared by different users or groups, and users may

belong to multiple groups. Second, encrypting the same content with different keys not only results in multiple redundant copies of the content in the cloud storage, but also diminishes the efficiency of content delivery via the distribution network.

Multicast security [4] aims to address the confidentiality of content sharing. However, in conventional multicast and broadcast settings, there are only two types of entities involved: multicast/broadcast center and users, and the center is the content provider or is fully trusted by the content provider. Their setting differs from our new cloud-based content delivery model, which requires a semi-honest cloud provider to assist the cloud provider and the users.

In cloud-based content storage and delivery services, the cloud provider provides two cloud-based services: *content storage service* and *content delivery network service*. By using them, the content provider is able to provide large-scale content sharing services to groups of subscribers through the public cloud. Subscribers consume the content by software installed on their host machines, such as a video player to play a digital movie file. It has been widely recognized that content security should be mainly relied on content providers who use the cloud-based services, instead of cloud service providers [5].

In this paper, we propose *CloudSeal*, an end-to-end solution for content confidentiality protection in the storage and delivery via cloud computing. By end-to-end, we mean that content is encrypted at cloud-based storage and delivery channels, and only authorized end users can decrypt it. We uniquely leverage several algorithms to achieve flexible security and efficient storage and distribution, including proxy re-encryption, k-out-of-n secret sharing, and broadcast revocation schemes. By proxy re-encryption algorithm, a content provider can transfer its initially encrypted content to the ciphertext so that only authorized subscribers can decrypt. To reduce the workload of content re-encryption from the content provider, a proxy is employed by the content provider to perform content re-encryption in the cloud. The content provider generates new re-encryption keys upon user joining or leaving.

In CloudSeal, when there is a request from a subscriber, the proxy service first checks if the content in the cloud storage is encrypted with the latest re-encryption key from the content publisher. If yes, the content can be downloaded via the content delivery service interface; otherwise, the proxy first invalidates any encrypted form of the target content via the delivery service, re-encrypts the content with the latest re-encryption key, and then authorizes the access. Therefore, there is only one encrypted copy of the content stored in cloud storage, and delivery network only serves contents encrypted with the latest re-encryption key. CloudSeal efficiently splits the ciphertext of the content into two parts. The re-encryption operation is only performed on a very small part, and the massive part remains unchanged. This feature enables efficient cache mechanism during content distribution.

The access control in CloudSeal is enforced by distributing a shared secret key to authorized users, with which re-encrypted content can only be decrypted. CloudSeal separates the distribution of the shared secret key from that of the

content and re-encryption keys. Therefore, it supports flexible authorization policies. Only authorized users can obtain the shared secret key, and the content provider maintains the control of issuing new keys whenever needed. CloudSeal leverages k-out-of-n secret sharing and broadcast revocation mechanism to renew the shared secret key to achieve scalability. Due to space limits, We refer the readers to the full version of our paper for more details on the implementation and evaluation of CloudSeal [6].

2 Model and System Goals

Threat Model Three types of parties are involved in our system: content provider, cloud provider, and subscriber. CloudSeal trusts the content provider and subscribers. Specifically, only the content rendering application or agent running on a subscriber's device is trusted, e.g., it does not release the content decryption key and any clear content to unauthorized parties, and it physically removes decryption key when the user leaves a group or is revoked by the content provider. We consider the cloud service provider to be *honest but curious* or semi-honest; that is, it follows the protocol and operations defined in CloudSeal, but it may actively attempt to gain knowledge of cleartext of the content. The content delivery service is also semi-trusted: it is curious to sniff content distributed and cached in the network, but it honestly performs all the operations and satisfies the quality of services, e. g., specified in service level agreement between the content provider and the delivery service provider. In addition, the cloud infrastructure (hardware and software) may be exploited by attackers who aim to expose the stored content [7].

We summarize the security and system objectives of CloudSeal as follows.

- CloudSeal should ensure data confidentiality when stored in cloud even under the collusion between the cloud provider and subscribers.
- CloudSeal should support dynamic system state, i. e., a user may choose to join or leave a group, or be revoked from a group by the content provider at any time.
- CloudSeal should support forward and backward security. For *backward security*, a user who leaves the group or is revoked from the group cannot access any data published after leaving or revocation. For *forward security*, a user cannot access any content that is published before she joins.

Beyond these security objectives, CloudSeal aims to achieve the following performance requirements. CloudSeal should preserve the efficiency of content delivery network. In particular, it is desirable for the network to store a single copy of encrypted content at each state for content integrity and distribution efficiency. Content decryption should not affect user experience at the device side, e. g., the speed of decrypting the video streaming should not be significantly lower than that of decoding.

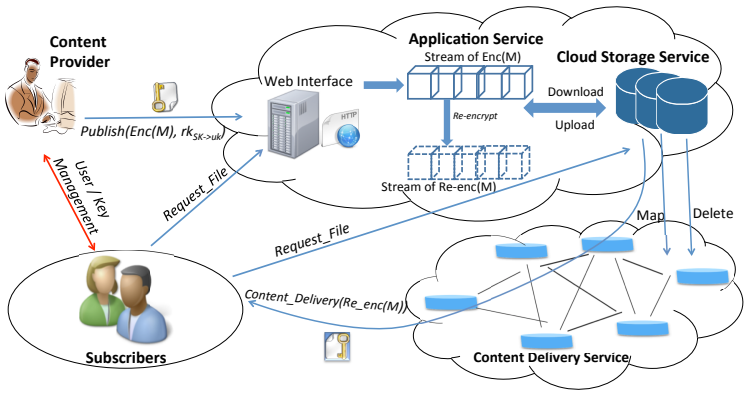

Fig. 1. CloudSeal overview.

3 CloudSeal Scheme Details

In this section, we first give an overview of CloudSeal, and then present details of the operations for content distribution and user management. Security properties of CloudSeal are then discussed.

3.1 Overview

Figure 1 shows the architecture of CloudSeal with three main parties: *cloud provider*, *content provider*, and *subscribers*.

- *Cloud Provider* provides two public cloud services: *storage service* for content storing and *content delivery network* for content distribution. It also provides virtual infrastructure to host application services, which can be used by the content provider to manipulate content stored in the cloud, or by content subscribers to retrieve content.
- *Content Provider* provides content to groups of subscribers, as well as user management. It uses cloud-based service from the cloud provider to store and distribute content.
- *Subscriber* is able to access to the content stored in the cloud if she successfully subscribes to the content provider. The subscriber can decrypt delivered content and consume it with local software.

Operations of CloudSeal are across two planes: data plane and control plane. In the *data plane*, we describe the implementations of content operations, including *system setup*, *content publishing*, *proxy re-encryption*, and *content retrieving*, along with involved cryptographic algorithms; in the *control plane*, we describe user management including *user subscription* – when a new user joins a group, and *user revocation* – when a user leaves or is revoked from a group. Our scheme utilizes the proxy re-encryption scheme proposed by Ateniese *et al.* [8] and the secret sharing scheme in [9].

3.2 Preliminary

Bilinear Maps [10, 11]: Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, we say e is a bilinear map if: (1) computative actions in $\mathbb{G}$ and $\mathbb{G}_T$ are efficient; (2) for all $\alpha, \beta \in \mathbb{Z}_r$ of prime order r, we have $e(g^\alpha, g^\beta) = e(g, g)^{\alpha\beta}$; (3) for any $g \in \mathbb{G}$, e is non-degenerate, i.e., $e(g, g) \neq 1$.

Secret Sharing [12, 9]: A k-out-of-n threshold secret sharing scheme is that a secret $S \in \mathbb{Z}_r$ shared by n users can be recovered, if the number of the secret shares exceeds the threshold k. The scheme utilizes a random polynomial P of degree $k-1$, where $P(x) \in \mathbb{Z}_r$ and $P(0) = S$. Given any k shares $< x_0, P(x_0) >$ $, ..., < x_{k-1}, P(x_{k-1}) >$, one can use Lagrange interpolation formulas as follows to recover $P(0)$:

$$P(0) = \sum_{i=0}^{k-1} \lambda_i P(x_i), \text{ where } \lambda_i = \prod_{j \neq i} \frac{x_j}{x_j - x_i} \tag{1}$$

Proxy Re-encryption [8]: A proxy re-encryption algorithm transforms ciphertext c_{k1} to ciphertext c_{k2} with a key $rk_{k1 \to k2}$ without revealing the corresponding cleartext, where c_{k1} and c_{k2} can only be decrypted by different key $k1$ and $k2$, respectively, and $rk_{k1 \to k2}$ is a re-key issued by another party, e.g., the originator of ciphertext c_{k1}.

3.3 CloudSeal Operations

Our cryptographic operations are described below. Our notation used in this paper is shown in Table 1.

Table 1. Notation.

Term	Notation	Term	Notation
PK	content provider's public key	P	polynomial formula
SK	content provider's secret key	x_i	user i's ID
uk	shared secret key for a group	$P(x_i)$	polynomial value of user i
$rk_{SK \to uk}$	re-encryption key	M	original content
k	number of shares to recover uk	h	a temporary secret of content provider

System Setup is run by the content provider to prepare the cryptographic system for content encryption and re-encryption. The content provider first chooses system public parameters $params$, namely $g \in \mathbb{G}$ and a bilinear map e. It chooses a secret key $SK \in \mathbb{Z}_r$ and public key $PK = g^{SK} \in \mathbb{G}$. The content provider keeps SK secret. This setup is performed by the content provider for each group of users. The content provider chooses an integer k and a list L of polynomials of degree $k-1$ with coefficients randomly chosen from $\mathbb{Z}_r$, which are kept secret. The number of users who can be revoked at the same time is $k-1$.

Content Publishing is followed by the content provider to publish its content to the public cloud. The content provider encrypts the content M before publishing it to public cloud with the secret key SK and $params$ as shown in Algorithm 1. The resulting encrypted content has two components (u_{SK}, v), both are stored in the content storage service by the application service via cloud APIs. u_{SK} depends on the random secret h and content provider's secret key SK, while v depends on both h and the content. Usually, u_{SK} is much smaller than v.

Algorithm 1: *Enc(params, M, SK)*

step1: Choose a random secret $h \in \mathbb{Z}_r$; let Z denote $e(g, g)$;
step2: Compute $Z^h = e(g, g^h) = e(g, g)^h \in \mathbb{G}_T$, $u_{SK} = g^{SK \cdot h}$; erase h;
step3: Output ciphertext of content M: $(u_{SK}, v) = (g^{SK \cdot h}, MZ^h)$.

Content Retrieving is for subscribers to access content stored in the public cloud. Two algorithms – *Re_Key* and *Re_Enc* – are involved in this process. The *Re_Key* algorithm is where content provider generates a content re-encryption key $rk_{SK \to uk}$ with its secret key SK and the current decryption key uk. Details are shown as follows.

Algorithm 2: *Re_Key(params, SK, uk)*

step1: Given $params, SK, uk$, the content provider computes $rk_{SK \to uk} = g^{uk/SK}$.

Upon request, the application service re-encrypts the target cipher content (u_{SK}, v) with the following *Re_Enc* algorithm.

Algorithm 3: *Re_Enc((u_{SK}, v), params)*

step1: Obtain the newest $rk_{SK \to uk}$ from the content provider;
step2: Calculate $u_{uk} = e(rk_{SK \to uk}, u_{SK}) = e(rk_{SK \to uk}, g^{SK \cdot h}) = e(g, g)^{uk \cdot h} = Z^{uk \cdot h}$;
step3: Output re-encrypted content (u_{uk}, v).

The application service stores u_{uk} in the content storage service and allows the download of the cipher content. u_{uk} and v can be cached in the content delivery network for download. The re-encryption is only performed on u_{SK}. Because u_{SK} is independent of the content M, CloudSeal saves the processing time and storage I/O cost between the application service and storage service.

When the system state is changed, i.e., the shared secret key is updated from uk to uk'. Once the new secret key is updated to authorized users (explained next), the content provider generates the re-key $rk_{SK \to uk'}$ by running *Re_Key* algorithm and sends the key to the application service for content re-encryption with *Re_Enc* algorithm. The new cipher content is $(u_{uk'}, v)$. The user can then download $u_{uk'}$ from the cloud storage service, and v from the content delivery network. u_{uk} is invalidated from the content delivery network by the application service before the download for backward security.

After a user obtains the encrypted content (u_{uk}, v), she follows Algorithm 4 below to decrypt the cipher with her current secret key uk. The user either obtains the secret key uk from the content provider when she first joins or computes it (described in User Subscription next).

Algorithm 4: *Decrypt((u_{uk}, v), uk)*

step1: Given u_{uk} and uk, compute $u_{uk}^{1/uk} = (Z^{uk \cdot h})^{1/uk} = Z^h$;
step2: Calculate $M = v/Z^h = (MZ^h)/Z^h = M$;
step3: Output original content M.

User Subscription happens when a user join the group. Successful subscription authorizes a user's access to protected content. To prevent a new user from accessing content published before joining (forward security), a new key and a share of secret are generated and distributed to the new user. To update remaining users' secret key, this share of secret is broadcasted to the group as follows. For the ease of description, we assume that $k = 2$ in what follows. Our algorithm can be generalized for any arbitrary k values.

- Upon receiving a join request from a new user, the content provider obtains the first polynomial $P' = ax + b$ on list L, and calculates key $uk' = P'(0)$; uk' is sent to the new user in a secure channel.
- The content provider assigns the new user a unique identity $x_i \in \mathbb{Z}_r$ and her share of secret from polynomial $P'(x_i)$, along with x_i's values for the other polynomials on list L. The content provider sends these polynomial values, except for $P'(x_i)$, to the new user for future key updating. P' is removed from the list L.
- The content provider broadcasts $< x_i, P'(x_i) >$ to the current group members for new key generation.
- For each current group member x_j, upon receiving $< x_i, P'(x_i) >$ from the content provider, she calculates the new key with her share of secret $P'(x_j)$ for P' that was received when x_j joined earlier. This user can recover the new secret key $uk' = P'(0) = b$ by calculating $P'(0) = \frac{x_j}{x_j - x_i} P'(x_i) + \frac{x_i}{x_i - x_j} P'(x_j)$ according to Equation 1.

User Revocation happens when a subscriber leaves a group or is revoked by the content provider. Our revocation scheme is based on k-out-of-n threshold secret sharing scheme.

- *Case I:* There are $k-1$ users to be revoked at one time. The content provider revokes $k-1$ users with shares $P(x_1), P(x_2), ..., P(x_{k-1})$, respectively. The content provider broadcasts the shares of secrets and identities of these users $< x_1, P(x_1) >, < x_2, P(x_2) >, ..., < x_{k-1}, P(x_{k-1}) >$ to the entire group. Each user x in the group combines her share of secret $< x, P(x) >$ with these $k-1$ shares, to interpolate the new secret key $uk' = P(0)$. The content provider uses uk' as the new shared secret key to generate re-encryption key for non-revoked users.
- *Case II:* There are t users to be revoked, where $t < k-1$. The content provider performs the revocation by sending the t shares of secret and additional $k - t - 1$ shares of the secret of polynomial P. These additional shares are values different from any existing users.

Polynomial P is then removed from the list L. If the list L is empty, the content provider adds new polynomials, as well as computes and distributes corresponding secret shares to current subscribers (for future interpolation purposes).

3.4 Security Analysis

Because published content is encrypted before being stored in the cloud storage service, and the system secret key SK is never released from the content provider, CloudSeal achieves the confidentiality of data in the public cloud. Furthermore, in any system state, with received re-encryption key $rk_{SK \to uk}$, the cloud service provider or an attacker cannot decrypt the cipher content. CloudSeal ensures that for any content access, the application service always uses the latest re-encryption key derived from the latest user secret key by the content provider, therefore only authorized users can decrypt the cipher content in any system state. By controlling the issuing of secret keys to authorized users, the content provider maintains the control of security policies.

Our re-encryption algorithm utilizes the proxy re-encryption proposed in [8], which has been proven to be secure against the Decisional Bilinear Diffie-Hellman Inversion problem. Besides, this re-encryption algorithm is resistant against collusion between revoked users and the cloud provider according to. This guarantees that the secret key of content provider is safe even either the user or the cloud provider obtains both re-encryption key and user's decryption key.

CloudSeal is able to protect content forward and backward security by integrating proxy re-encryption and k-out-of-n secret sharing scheme. When an user joining or leaving event happens in a user group, CloudSeal clears old content stored in content delivery network and alters content to be delivered with updated decryption key. Therefore, new users can not decrypt the old content by the new key; revoked users can not decrypt the new content with their old keys.

Leveraging content delivery network, CloudSeal uniquely achieves content protection and distribution efficiency. When the system state changes, only a small part of a cipher content needs to be re-encrypted, such that most of the content object can be cached in cloud and shared by users. The separation of content operations (data plane) and user management operations (control plane) further enables flexible and scalable deployment of CloudSeal in the cloud and highly distributed environment.

4 Related Work

Several security solutions have been recently developed for securing the cloud [13, 14, 15, 16], including secure data access, data privacy, and operations on encrypted data. With similar security concerns in cloud service, Yu et al. [15] proposed an attribute based access control policy to securely outsource sensitive user data to the cloud. CloudSeal is different from their approach in that: CloudSeal only allows a content provider to perform the Re_Key operation, and our proxy re-encryption is performed directly on part of the cipher content. Therefore, directly applying their approach in the problem that we target here is not practical, as their ciphertext data is customized for different users. Essentially, efficient data distribution with common ciphertext that can be cached in content delivery network is not the goal of [15].

Secure storage system is an important application of proxy re-encryption [8, 17]. CloudSeal is based on the scheme proposed in [8], where the authors describe an encrypted file storage with an access control server in charge of data access according to their proxy re-encryption methods. In comparison, we deploy the re-encryption algorithm in a unique cloud-based content delivery application. CloudSeal also supports the k-out-of-n secret sharing for efficient user-revocation purposes.

Secure multicast communication [18, 4, 19, 20] and conditional access systems [21] address similar security problems as ours in distributing content to dynamic user groups and key management. Proxy re-encryption and k-out-of-n mechanisms are also used to solve these problems. The problem solved by CloudSeal is different from them due to the cache properties in content delivery network, which requires more efficient and flexible secure content delivery and user management mechanisms.

5 Conclusion and Future Work

We design CloudSeal, an end-to-end content confidentiality protection mechanism for large-scale content storage and distribution systems over public cloud infrastructure. By leveraging advanced cryptographic algorithms including proxy re-encryption, threshold secret sharing, and broadcast revocation, CloudSeal addresses unique challenges of efficient cipher content transformation, cipher content cache in delivery network, and scalable user and key management. We have implemented a prototype of CloudSeal based on Amazon EC2, S3, and CloudFront services. Our initial evaluation results demonstrate that CloudSeal can provide efficient and scalable secure content storage and delivery in cloud-based storage and content delivery network. The details of our implementation and evaluation can be found in [6]. For future work, we plan to investigate practical and scalable browser-based methods for distributing secret information from the content provider to the subscribers.

References

1. Cloud Computing, an IDC update `http://www.slideshare.net/JorFigOr/cloud-computing-2010-an-idc-update`, 2010.
2. Yunhua Koglin, Danfeng Yao, and Elisa Bertino. Secure Content Distribution by Parallel Processing from Cooperative Intermediaries. *IEEE Transactions on Parallel and Distributed Systems*, 19(5):615–626, 2008.
3. Danfeng Yao, Yunhua Koglin, Elisa Bertino, and Roberto Tamassia. Decentralized Authorization and Data Security in Web Content Delivery. In *Proc ACM Symp. on Applied Computing (SAC)*, pages 1654 – 1661, 2007.
4. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast Security: A Taxonomy and Some Efficient Constructions. In *Proceedings of INFOCOM*, March 1999.
5. AWS Customer Agreement `http://aws.amazon.com/agreement/`, 2011.

6. Huijun Xiong, Xinwen Zhang, Wei Zhu, and Danfeng Yao. CloudSeal: End-to-End Content Protection in Cloud-based Storage and Delivery Services. Technical report, Huawei Research, 2011.
7. Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, You, Get Off of My cloud! Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of ACM Conference on Computer and Communications Security*, 2009.
8. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. *ACM Trans. Inf. Syst. Secur.*, 9:1–30, February 2006.
9. Moni Naor and Benny Pinkas. Efficient Trace and Revoke Schemes. In *Proceedings of the 4th International Conference on Financial Cryptography*, FC '00, 2001.
10. Dan Boneh and Matthew K. Franklin. Identity-based Encryption from the Weil Pairing. In *Proceedings of CRYPTO*, 2001.
11. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT, 2001.
12. Adi Shamir. How to Share A Secret. *Commun. ACM*, 22, November 1979.
13. Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Authorized Private Keyword Search over Encrypted Personal Health Records in Cloud Computing. In *Proceedings of The 31st Int'l Conference on Distributed Computing Systems (ICDCS 2011)*, 2011.
14. Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In *Proceedings of INFOCOM*, 2010.
15. Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *Proceedings of INFOCOM*, 2010.
16. Saman Zarandioon, Danfeng Yao, and Vinod Ganapathy. K2C: Cryptographic Cloud Storage With Lazy Revocation and Anonymous Access. In *Proceedings of Securecomm*, 2011.
17. Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable Secure File Sharing on Untrusted Storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, Berkeley, CA, USA, 2003.
18. Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure Group Communications Using Key Graphs. *IEEE/ACM Trans. Netw.*, 2000.
19. Bob Briscoe. MARKS: Multicast Key Management using Arbitratily Revealed Key Sequences. In *Proceedings of 1st International Workshop on Networked Group Communication (NGC'99)*, 1999.
20. Bob Briscoe. Nark: Receiver-based Multicast Non-repudiation and Key Management. In *Proceedings of ACM Conference on Electronic Commerce (EC'99*, 1999.
21. Patrick Traynor, Kevin R. B. Butler, William Enck, and Patrick McDaniel. Realizing Massive-Scale Conditional Access Systems Through Attribute-Based Cryptosystems. In *NDSS*, 2008.