k-times Anonymous Authentication

NEC Corporation

Isamu Teranishi, Jun Furukawa, and Kazue Sako

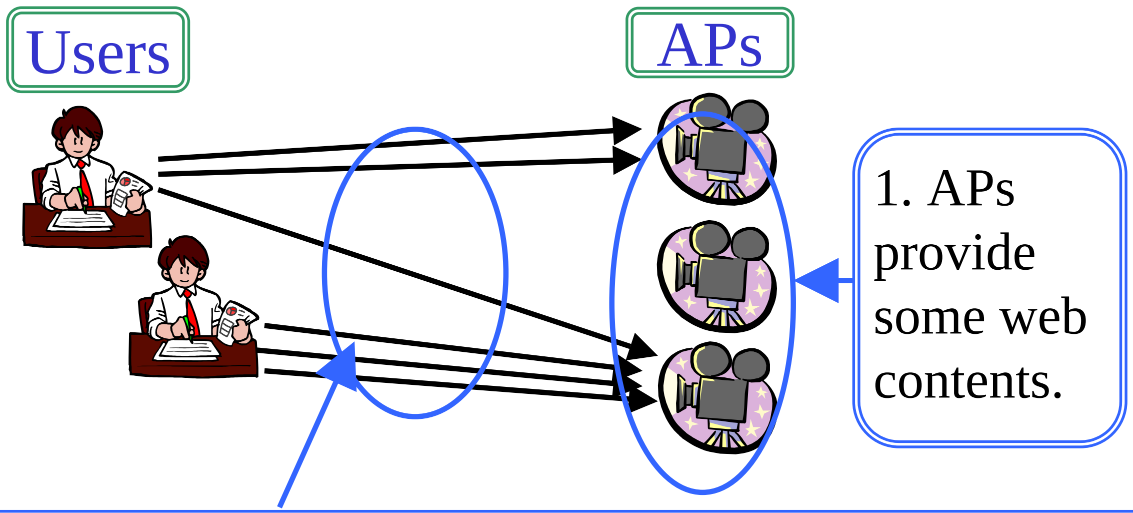
Table of Contents

Motivation (1)

Many Applications need to restrict
the number of times user can access to a service.

E-Cash E-Coupon
E-Voting Trial Browsing …

Trial Browsing



2. Users can browse the contents for free on trial for only a restricted number of times.

3. If a user has browsed beyond the restricted times, the over-times user should be identified, so that an AP can send a bill to him.

Motivation (2)

Requirements of Trial Browsing

1. Each AP can independently determine the number of times, a user is allowed to access.
2. To protect user's privacy, a user who had browsed only within a given number of times will remain to be anonymous.
3. To prevent malicious users from browsing over-times, a user who had browsed beyond a given number of times can be identified.

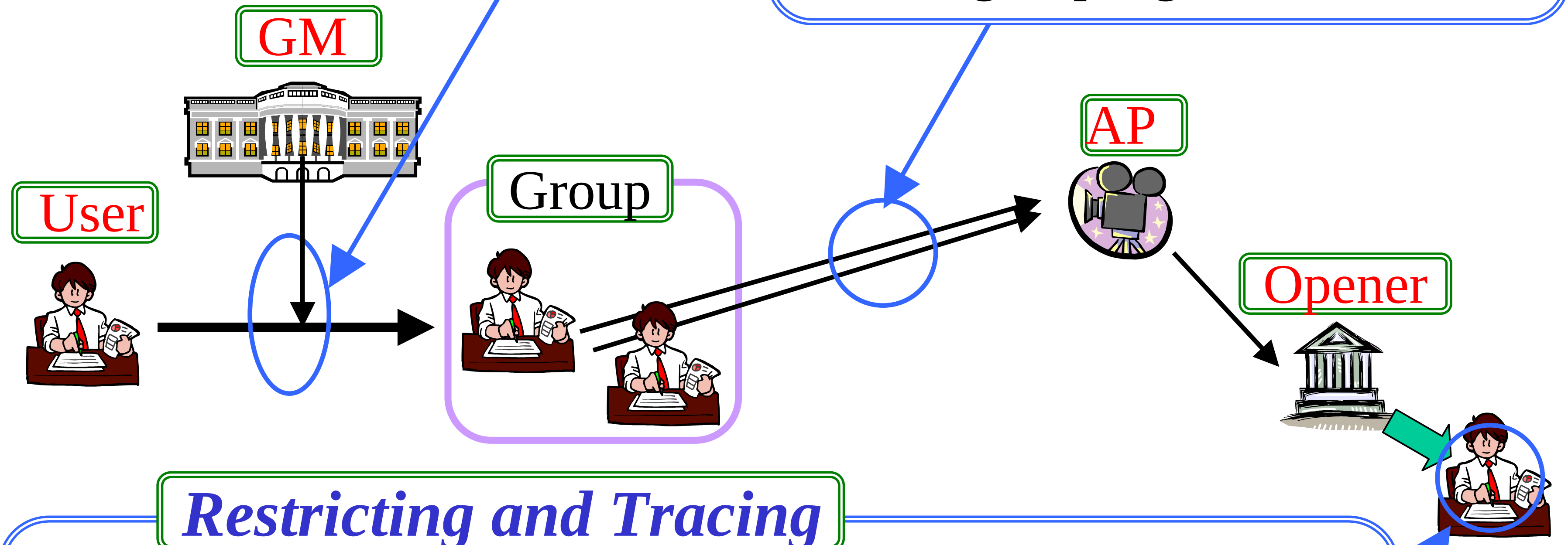
A Trivial Solution using Group Signature

Joining

1. The GM registers a user who wants to browse web contents.

Authentication

2. When a member of the group browses contents, the member issues a group signature to the AP.



Restricting and Tracing

3. The opener identifies the user who is authenticated, records he accessed to which AP and then checks and reports the AP whether the user has been authenticated more than the allowed number of times or not.

Problems in the Trivial Solution

1. The opener is able to trace even honest users.
   - We want to provide privacy to honest users as much as possible. Therefore, it is desired that no one can trace honest users.
2. The opener needs to involve in all authentication of all APs and count the number of time users have accessed to each AP.
   - This is very cumbersome.
   - If the opener is unaccessible, all AP cannot count the number of times users have accessed. Hence, the APs must stop services.

Our Goal

To construct a scheme which satisfies the following properties:

1. Each AP can independently determine the number of times, he allows users to access to him.
2. There is no authority who can identify an honest user who has been authenticated only within an allowed number of times.
3. There is no opener. Anyone can identify a dishonest user who has been authenticated beyond an allowed number of times from the authentication log without help of any authorities.

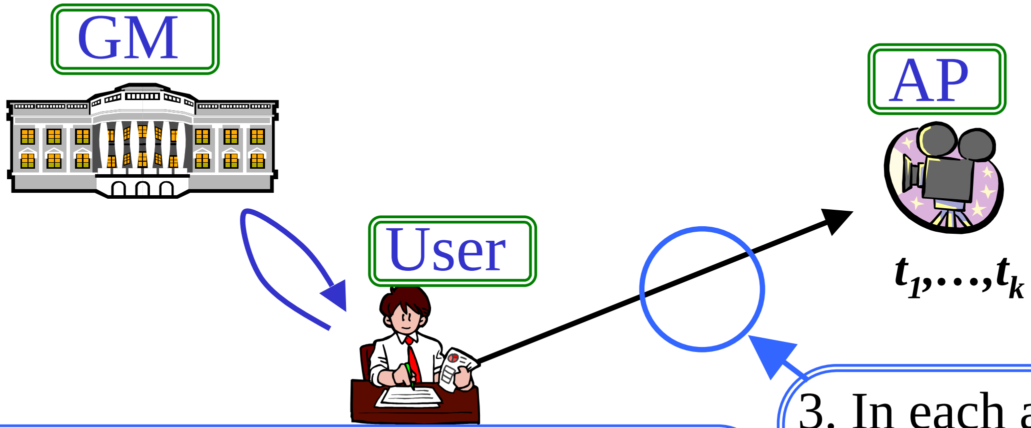
Road Map until We Construct the Scheme

We construct the proposed scheme by modifying the trivial solution.

1. We first modify the trivial solution into a naive scheme in such a way that the identification of members by the opener is impossible, and that the number of times user can be authenticated will be restricted.
2. Next, we add a mechanism to the naive scheme. This mechanism enables anyone to identify the user who has been authenticated for more than an allowed number of times.
3. Finally, we refine the above scheme to achieve stronger anonymity.

Naive Scheme

We introduce a naive scheme which can restrict the number of times users can be authenticated by each AP.



1. As in the case of the trivial solution, the GM generates user's membership certificate/secret key pair. We let $\boldsymbol{x}$ be the user's secret key.

2. Each AP publishes randomly generated $\boldsymbol{k}$ ***search tag bases*** $t_1,\ldots,t_k$. Here, $\boldsymbol{k}$ is the number of times that the AP allows users to access.

3. In each authentication, a user computes a ***search tag*** $\tau = t_i^x$, using his secret key $\boldsymbol{x}$ and one of search tag bases t_i which he has not used before. Then the user prove the validity of the search tag.

Since there are only $\boldsymbol{k}$ search tag bases, if a user has been authenticated for more than $\boldsymbol{k}$ times, the user must have used one of the search tag bases, t_i , at least twice. Hence, in such a case, anyone can find search tags τ and ν which satisfy $\tau = \nu \ (=t_i^x)$. Therefore, anyone can decide whether some user is authenticated beyond allowed number of times.

Tracing Tag Mechanism

Search tag mechanism only helps an AP to find the incident that some user has accessed for more than an allowed number of times, but this user cannot be identified. Hence, we introduce a `` ***tracing tag mechanism*** '' that helps tracing of such a user.

1. In advance, the GM publishes random ***b***, and each user publishes $\beta = b^x$. (***x*** is the secret key of the user).
2. An AP publishes randomly generated ***tracing tag bases*** $t'_1, \ldots, t'_k$, additionally to the search tag bases $t_1, \ldots, t_k$.
3. In each authentication of a user, the AP first sends randomly selected ***l*** to the user. Then the user computes a ***tracing tag*** $\tau' = b^{xl} t'^{x}_{i}$, and sends a pair of search tag τ and tracing tag τ', (τ, τ'), to the AP.

Identifying a Malicious User (1)

From the property of the search tag mechanism, if a malicious user has accessed some AP more than an allowed number of times, one can find pairs of search tag and tracing tag,

$$(\tau, \tau') \text{ and } (\nu, \nu')$$

which satisfies $\tau = \nu$.

This is because, from search tag mechanism, the following equations hold for some ***i***

$$(\tau, \tau') = (t_i^x, b^{xl}\, t'^x_i),$$
$$(\nu, \nu') = (t_i^x, b^{xl^*} t'^x_i).$$

Identifying a Malicious User (2)

Hence, the following equation is satisfied:

$$\tau' / \nu' = (b^{xl}\ t_i^{x})/(b^{xl^*}\ t_i^{x}) = b^{xl} / b^{xl^*} = b^{x(l-l^*)} = \beta^{(l-l^*)}$$

Therefore, it follows

$$\beta = (\tau' / v')^{\frac{1}{(l-l^*)}}. \qquad \cdots\ (1)$$

Using equation (1), anyone can compute β from **(τ', ν', l, l*).**

The pair **(τ', ν', l, l*)** is written in the authentication log. Since β is published by the malicious user, anyone can identify the malicious user using only public information and the authentication log.

Anonymity (1)

The naive scheme with tracing mechanism does not provide users enough privacy. We show two threats to obtain some user related information. Then, we introduce refinements of the scheme.

First Threat

Recall that a search tag τ satisfies $\tau = \boldsymbol{t_i^x}$, where $\boldsymbol{x}$ is some user's secret key and $\boldsymbol{t_i}$ is a search tag base.

Suppose search tag bases satisfy some known relation, such as $\boldsymbol{t_i = t_j^2}$.

Then the corresponding search tags τ_i and τ_j satisfy $\tau_i = \tau_j^2$, if and only if the τ_i and τ_j are generated by the same user.

Therefore, an AP can decide whether the same user is authenticated.

Refinement

We force each AP to set a pair of search and tracing tag bases, $(\boldsymbol{t_i, t'_i})$ = Hash(some data). Then, in each authentication, a user verifies that the pair is a correct hash value.

Anonymity (2)

Recall that an honest user **U** uses each pair of search and tracing tag bases, (t_i, t'_i), only for once.

Hence, if user **U** had ever used a pair (t_i, t'_i), he will not use this pair again.

Suppose user **U** already used a pair (t_i, t'_i).

Second Threat

Then if an authenticated user used the pair (t_i, t'_i), an AP can decide that the user is not **U**.

Refinement

In each authentication, a user does not reveal to an AP which pair (t_i, t'_i) the user uses, and prove the knowledge of pair (t_i, t'_i) in zero-knowledge.

Security Definitions

Total Anonymity

Even the GM can not decide two authentications are performed by same user or not, if the user(s) is/are honest.

Detectability

Each user cannot be accepted more than allowed number of times without help of the GM.

Exculpability for user

Honest tracing procedure does not output the ID of an honest users.

Exculpability for the GM

Honest tracing procedure does not output the ID of the GM if the GM is honest.

Formal definition of these requirements are given in our paper.

Assumptions

We can prove that our scheme satisfies the previous requirements under the following assumptions:

- The strong RSA assumption
- The DDH assumption
- The existence of random oracle
- The existence of PKI.

We must assume the existence of infrastructure that can guarantee the correspondence between a user and his public information β. PKI is only an example of such an infrastructure.

In our paper, instead of PKI, we have newly proposed and assumed simpler infrastructure ``List Oracle’’.

Applications

Trial Browsing Scheme

With our scheme, trial browsing with easy tracing of malicious users and strong anonymity for honest users is possible.

E-Coupon Scheme

With our scheme, the number of times users can use coupons can be restricted. The scheme achieves both easy tracing of malicious users and strong anonymity for honest users.

E-Voting Scheme

If we restrict the number of times each user can vote by our scheme, in an e-voting scheme, we can conceal whether each user has voted or not.

Conclusions

- We proposed a scheme each user can be authenticated anonymously within an allowed number of times.
- The scheme satisfies the followings:
  1. Each AP can independently determine the number of times, he allows users to access to him.
  2. There is no authority who can identify an honest user who has been authenticated only within an allowed number of times.
  3. Anyone can identify a dishonest user who has been authenticated beyond an allowed number of times from the authentication log without help of any authorities.
- We formalized the security requirements.
- The scheme can be applied to trial browsing of contents, e-coupon and e-voting.

Fin.