The following paper was originally published in the
Proceedings of the 3rd USENIX Workshop on Electronic Commerce
Boston, Massachusetts, August 31–September 3, 1998

General-purpose Digital Ticket Framework

Ko Fujimura and Yoshiaki Nakajima
NTT Information and Communication Systems Labs

For more information about USENIX Association contact:

1. Phone: 1 510 528-8649
2. FAX: 1 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

General-purpose Digital Ticket Framework

Ko Fujimura and Yoshiaki Nakajima
NTT Information and Communication Systems Labs
{fujimura, yoshiaki}@isl.ntt.co.jp

Abstract

A digital ticket is a certificate that guarantees certain rights of the ticket owner. There are many applications for digital tickets but the ticket properties vary depending on the application. This variety makes the digital ticket processing system expensive, especially if dedicated systems must be developed for each application. This paper thus addresses issues on developing a common data schema and processing architecture for various types of digital tickets. This paper clarifies requirements for a general-purpose digital ticket and shows four features in contrast to digital cash: 1) parameterization of ticket properties on anonymity, transferability, and divisibility; 2) machine-understandability of ticket contents; 3) state-transitionality of ticket status; and 4) composability of multiple tickets. To achieve parameterization of ticket properties and machine-understandability, we propose a Resource Description Framework (RDF)-based ticket description method. Its metadata facility enables various ticket properties to be defined using multi-layered schemata. To achieve state-transitionality and composability, we propose describing a ticket using a set of signed descriptions linked with restriction-specified incomplete/complete links. Finally, this paper proposes a set of common ticket processing components.

1. Introduction

A number of electronic payment schemes [1] such as encrypted credit cards [17], digital cash [8], and micropayments [5] [16] have been designed and established for Internet commerce. However, in the opposite flow of the payment, i.e., goods or products to the consumer, we depend on a physical delivery system except for a few types of digital contents, e.g., images, sounds, and computer software. Any goods that can be encoded as digital information, however, can inherently be delivered electronically.

Any business that offer services, e.g., transportation, accommodations, theaters, and restaurants, also make up a large industry and their services can be sold as tickets. A ticket is a certificate, which guarantees that the ticket owner has the right to claim the services written on the ticket. A ticket can be implemented as a digital certificate (or digital ticket) and can be delivered electronically. Although they are not common now, the business scope of electronic commerce will expand rapidly if digital tickets, which enable us to trade over the Internet, become easy to use and manage.

The number of any one particular type of digital ticket that is issued, however, would be far less compared to digital cash, because there is such a wide variety of tickets. It makes the implementation cost of the infrastructure for processing digital tickets, such as ticketing systems, ticket wallet systems, and ticket examination systems, expensive, if a system must be developed for each individual application. This paper thus addresses a ticket description method and processing architecture of general-purpose digital tickets that enable the issuing, trading, and spending of various types of digital tickets using a set of common ticket processing components.

Some digital tickets, e.g., E-Stamp [12] or e-gold [13], have already been developed. Transaction Net [19] surveys several digital values including commodity-backed money or scrip. These technologies, however, were developed for individual tickets or applications. No digital ticket framework that covers a wide range of digital tickets has been proposed yet.

This paper clarifies the requirements of general-purpose digital tickets and its four features which are not required for digital cash: 1) parameterization of ticket properties on anonymity, transferability, and divisibility; 2) machine-understandability of ticket contents; 3) state-transitionality of ticket status; and 4) composability of multiple tickets.

To establish parameters for properties and machine-understandability, this paper defines layered schemata of digital tickets using Resource Description Framework (RDF) [10]. Its metadata facility enables various ticket properties to be defined and enables the use of various ticket schemata according to the ticket type such as theater tickets or accommodation tickets. To achieve state-transitionality and composability, we propose describing a ticket using a set of signed descriptions linked with restriction-specified links. The state-transition of a ticket, e.g., payment status or re-

servation status, is expressed by attaching to the original ticket a description of how the status was changed. Using the restriction-specified links, flexible requirements on the description to be attached can be specified.

Finally, this paper proposes a set of common ticket processing components that can issue, trade, and spend various types of digital tickets.

2. Digital Ticket

In this paper, a digital ticket (simply called "ticket" hereafter) is defined as follows:

Definition

Let *I* be a ticket issuer, *O* be a ticket owner, and *P* be a promise to the ticket owner. A ticket is defined as Signed_I (*I*, *P*, *O*), where the phrase "Signed_I" means that the entire block is signed by the issuer's digital signature.

Examples of promise *P* are as follows:

- A flight between Boston and Tokyo can be reserved with this ticket.
- This ticket can be exchanged for 1g of gold.
- One image file in a particular server can be downloaded with this ticket.
- After June 1998, this ticket can be exchanged for my car.
- The bearer of this ticket has unlimited telephone use for one month.

In this paper, we assume the issuance, transference, and consumption of a ticket are as follows:

Ticket Issuance

A ticket can be freely issued bearing the ticket issuer's intention, e.g., service to be provided or commodity to be delivered. It represents a promise by the ticket issuer to complete some task or render some service. If the task described on the ticket cannot be accomplished, it will detract from the ticket issuer's credibility.

Ticket Transference

Depending on the type of ticket, as described in Section 3, a ticket can be transferred to a third person. Typically, transferring a ticket can be accomplished using a special software component described in Section 4.4. Note that payment is usually also made for the transference of the ticket, but this payment is outside the scope of this paper. We make no assumption on the payment method.

Ticket Consumption

A ticket is consumed typically by the issuer fulfilling the service or task represented by the ticket and the ticket being returned to or voided by the issuer or service provider. More concretely, the mechanism of this process can be represented by placing a ticket in a ticket examination machine when a service is rendered. Another example is pasting a ticket icon in an input field of a form received from some service provider (downloading an image file, etc.) to receive a service. In this process, it is assumed that the consumer agreed to void the ticket. A certificate of the consumption signed by the consumer is given to the issuer or service provider.

3. Related Work

Digital tickets have many similarities to digital cash. Therefore, some basic technologies for implementing digital tickets have already been developed in the research area of digital cash [8]. This section thus compares and contrasts the digital ticket to digital cash as we know it today. The following ten key features [6][7] of digital cash have been proposed.

(1) Secure (unable to alter or counterfeit)
(2) Anonymous (untraceable)
(3) Portable (physical independence)
(4) Transferable
(5) Off-line capable
(6) Divisible [7]
(7) Infinite duration (persistent)
(8) Wide acceptability (trust)
(9) User-friendly (easy-to-use)
(10) Monetary freedom (non-political) [6]

As a result of our investigation on many physical tickets, we found that required levels on (2) anonymity, (4) transferability, (6) divisibility, and (7) persistency are different according to the ticket type as shown in Table I and II. It is important for a general-purpose digital ticket architecture to process any type of ticket regardless of the required property level. Thus establishing parameters for these properties is required to cover a wide range of tickets.

In addition, we found that the three following requirements are important for digital tickets, which are not required in digital cash.

(11) Machine-understandable

Before any transaction can be conducted, the terms and description of the service or task must be objectively understood by both the service provider and consumer or owner, otherwise, the value of the ticket can not be determined. Moreover, as described in Section 4.4, this is a key property to register or search for a ticket in a marketplace when the ticket is resold.

(12) State manageable

Some tickets have a payment status, i.e., paid or unpaid, and/or reservation status, e.g., waiting list, reserved, or canceled. The status may be changed dynamically. Additionally, the ticket owner's identification that is recorded on the ticket can be rewritten when the ticket is transferred. However, it is difficult to allow these change while still guaranteeing security.

(13) Composable

Combining two or more tickets is sometimes required to obtain a service or one ticket may comprise several parts. For example, a travel ticket can comprise an accommodation ticket and a plane ticket. Also, a plane ticket can comprise a flight reservation ticket and an open ticket.

As mentioned above, digital tickets have some similar properties to digital cash but these properties vary with the ticket, and there are some new requirements not required for digital cash. A summary of this is shown in Table I.

Among the technologies to achieve the above ten properties, (1) through (10) are outside the scope of this paper since a number of methods [1][8] have already been proposed and developed. Instead, this paper proposes a ticket description method that enables the definition of various ticket properties and first establishes (11) a machine understandable ticket. Second, this paper proposes a ticket model that achieves (12) state manageable and (13) composable tickets. Finally, this paper briefly explains the prototype system of common ticket processing components that we developed.

4. Approach

4.1 Ticket description

Basic properties of a ticket comprise issuer *I*, promise *P*, and owner *O* as described in Section 2.

Properties		Digital Cash	Digital Ticket
(1)	Secure	Yes	Yes
(2)	Anonymous (Untraceable)	Yes	(2-1) Untraceable (2-2) Traceable
(3)	Portable (Physical independence)	Yes	Yes
(4)	Transferable	Yes	(4-1) Transferable (4-2) Not transferable
(5)	Off-line capable	Yes	Yes
(6)	Divisible (Number of times to be consumed)	Yes	(6-1) Specified times (6-2) Only once (6-3) Infinite times
(7)	Persistent (Valid period)	Yes	(7-1) Specified period (7-2) Persistent
(8)	Wide acceptability	Yes	Yes
(9)	User-friendly	Yes	Yes
(10)	Monetary freedom	Yes	No
(11)	Machine-understandable	No	Yes
(12)	State manageable	No	Yes
(13)	Composable	No	Yes

Table I. Digital Ticket Properties

Examples	Anonymity[1]	Transferability[1]	Number of times
Event ticket	Yes	Yes	Only once
Plane ticket	No	No	Only once
Lottery ticket	Yes	No	Only once
Stamp	Yes	Yes	Only once
Telephone card	Yes	Yes	Specified
Cash	Yes	Yes	Specified
Software license	No	Yes	Infinite
Transportation pass	Yes	No	Infinite
Gate card	No	No	Infinite
Driver's license	No	No	Infinite

[1]It depends on the specific ticket. This table only shows the tendency for the ticket types.

Table II. Properties of Specific Ticket Types

Transferability, anonymity, number of times to be consumed, and valid period shown in Table I are also important properties that vary depending on the ticket and determine how the ticket should be processed. Regarding the anonymity property, however, it can depend on how owner *O* is specified. For example, anonymity can be achieved by using a pseudonym [3][4] to specify the owner and keeping the real name secret except to the pseudonym issuer. We therefore assume that the anonymity property is subsumed by the owner property. To display or print the ticket, a view property is also needed. All of these properties can be and must be specified regardless of the ticket type.

There are properties determined by each ticket type, e.g., event ticket, plane ticket, accommodation ticket, and software license. Examples of these properties are

flight number or departure date, etc. These properties are defined by each industry.

In addition, there are properties that are defined by each issuing company or individual. An example of these properties is mileage points. Ticket properties are thus classified into the following three schema layers.

Layer 1
Common ticket properties that do not depend on the ticket type:

- Issuer
- Promise (Details are defined in upper layers)
- Owner (incl. pseudonym)
- Transferability
- Number of times to be consumed
- Valid period
- View
- Issuer's signature on above

Layer 2
Common ticket properties defined by each industry

Layer 3
Any ticket property defined by each issuing company or individual

We examined existing data describing technologies to see if they can be used to describe various ticket properties and if they can use multi-layered schemata. We found that Resource Description Framework (RDF) [10] developed by W3C fulfills the requirements above. RDF is a foundation for exchanging machine-understandable information on the Web and no application-specific assumption is made. We therefore defined ticket-specific schema layers based on RDF.

Figure 1 shows an example of describing a concert ticket in the RDF syntax. Note that the RDF specification is a draft specification and may be updated or replaced.

In RDF, the meaning and restrictions of the properties used in an RDF description must be defined in the RDF schema, which are defined somewhere in the network. The RDF schema is defined in an RDF schema specification [11]. See details in the specification.

Using XML [9] namespace facility, which may be contained in the RDF document's prolog, Universal Resource Identifiers (URIs) for layered ticket schemata can be referred to as described in this example. In this example, it is assumed that each schema is distributed and their URIs are defined as follows:

Layer 1
```
<?xml:namespace
   name="http://tickets.org/schemas/ticket#"
   as="DTK"?>
```

Layer 2
```
<?xml:namespace
   name="http://events.org/schemas/event-
   ticket#" as="EVT"?>
```

Layer 3
```
<?xml:namespace
   name="http://mycorp.com/schemas/my-ticket#"
   as="MTK"?>
```

This facility makes it easy to specify and maintain the schemata defined by different organizations. This approach, thus, not only achieves (11) the machine-understandability feature described in Section 3, but also has the advantage that many organizations can easily define their own ticket schema.

As shown in Figure 1, the `<DTK:Issuer>`, `<DTK: Promise>`, `<DTK:Owner>`, `<DTK:Transferability>`, `<DTK: NumberOfTimes>`, `<DTK:ValidPeriod>`, `<DTK:View>`, and `<DTK:Signature>` tags are used to define the layer 1 properties.

The Conditions property in Figure 1 describes detailed contract conditions. Such information is not always necessary but the size is not negligible for circulating a ticket, especially when the ticket is recorded on a smart card. In this case, the description can be located on a certain server on the network and referred to by the link if necessary. It also reduces communication cost. The View property, which defines the image data of the ticket, is also defined using a link as shown in Figure 1.

In RDF, the Description element itself creates a resource and it can be referred to by the bagID. Using the bagID, the scope of the definition that the signature is applied to can be specified. See details in the specification [10].

4.2 Restriction-specified incomplete link

In this section, we propose an approach to implement the (12) state-manageable ticket features. State-manageability enables the status of the ticket, e.g., owner identifier, payment status, or reservation status, to change dynamically.

```
<?xml:namespace name="http://mycorp.com/schemas/my-ticket#" as="MTK"?>
<?xml:namespace name="http://events.org/schemas/event-ticket#" as="EVT"?>
<?xml:namespace name="http://tickets.org/schemas/ticket#" as="DTK"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>

<RDF:RDF>
  <RDF:Description ID="ticket_001" bagID="ticket_bag_001">
     <DTK:Issuer>issuer@mycorp.com</DTK:Issuer>
     <DTK:Promise>
        <RDF:Description>
           <EVT:Reference>R02-345</EVT:Reference>
           <EVT:Name>FooBar Concert</EVT:Name>
           <EVT:Type>Adult</EVT:Type>
           <EVT:Place>New York City Hall</EVT:Place>
           <MTK:Points>20</MTK:Points>
           <MTK:Conditions href="http://mycorp.com/conditions"/>
        </RDF:Description>
     </DTK:Promise>
     <DTK:Owner>u1@host1.com</DTK:Owner>
     <DTK:Transferability>ANYBODY</DTK:Transferability>
     <DTK:NumberOfTimes>ONCE</DTK:NumberOfTimes>
     <DTK:ValidPeriod>1998-2-20</DTK:ValidPeriod>
     <DTK:View href="http://mycorp.com/my-ticket.gif"/>
  </RDF:Description>

  <RDF:Description href="#ticket_bag_001">
     <DTK:Signature>05b8cfc04d05a8cfc03d2549cfc03d36</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

Figure 1. A concert ticket example in RDF

However, it is not easy to implement because the ticket is signed by the issuer, and nobody except the issuer can alter the contents of the ticket. However, it is often required to change the contents of the ticket with the approval of the original ticket issuer or a Trusted Third Party (TTP), etc.

To express the state-transition of ticket properties without changing the original ticket definition, we propose defining a ticket using a linked set of signed descriptions, i.e., the state-transition of a ticket property is defined by attaching a state-transition description linked from the property definition in the original description:

Original Description:

```
<RDF:RDF>
  <RDF:Description ID="ticket_001"
                  bagID="ticket_bag_001">
    ...
    <Prop1/>
      <RDF:Description>
         <DTK:Value>current-value</DTK:Value>
         <DTK:NewValue href="#prop1_001"/>
      </RDF:Description>
    </Prop1>
    ...
  </RDF:Description>
  <RDF:Description href="#ticket_bag_001">
    <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

Assume that the value of property Prop1 is changed from "current-value" to "new-value," then the following description is attached to the above:

Attached Description:

```
<RDF:RDF>
  <RDF:Description ID="prop1_001"
                   bagID="prop1_bag_002">
     <DTK:Value>new-value</DTK:Value>
  </RDF:Description>
  <RDF:Description href="#prop1_bag_002">
     <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

In the above examples, link ID "prop1_001" must be defined and signed before the referred description is defined. We call such a link an *incomplete link*.

The link ID must be universally unique, since the attached signed description could be used for malicious purposes. An URI can be used for that purpose. Note that short IDs are used in this paper for readability.

The incomplete link shown in the above example, allows any description to be located as the destination of the link without restriction. It is inconvenient especially when the referred description must be signed (or issued) by the original issuer or others who have been delegated the right to define the value. To fulfill this requirement, this paper proposes a new concept called

restriction-specified incomplete link that restricts only the valid description so that it can be located on the destination of the incomplete link. If an invalid description is located, it will be detected by the validation check system.

An example of a restriction-specified incomplete link is as follows:

Original Description:

```
<RDF:RDF>
  <RDF:Description ID="ticket_001"
                   bagID="ticket_bag_001">
    ...
    <Prop1>
      <RDF:Description>
        <DTK:Value>current-value</DTK:Value>
        <DTK:NewValue href="#prop1_001"/>
        <DTK:Restriction>
         <RDF:Description>
          <DTK:Issuer>u2@host.com</DTK:Issuer>
         </RDF:Description>
        </DTK:Restriction>
      </RDF:Description>
    </Prop1>
    ...
  </RDF:Description>
  <RDF:Description href="#ticket_bag_001">
     <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

This example restricts the description of prop1_001 to be attached to the description of issuer u2@host.com. An example of the valid description is as follows:

Attached Description:

```
<RDF:RDF>
  <RDF:Description ID="prop1_001"
                   bagID="prop1_bag_002">
    <DTK:Issuer>u2@host.com</DTK:Issuer>
    <DTK:Value>new-value</DTK:Value>
  </RDF:Description>
  <RDF:Description href="#prop1_bag_002">
    <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

As we described in the above examples, a restriction-specified incomplete/complete link is defined as follows:

Definition

Let D_0 be a signed description, P be a property in D_0, V be the current value for P, D_1 be a signed description to be attached, L be a link to D_1, and R be a restriction for D_1. A *restriction-specified incomplete/complete link* is defined as a tupple of (P, V, L, *and* R) .

The value of P is interpreted as D_1 if D_1 is instantiated and D_1 satisfies restriction R, otherwise the value of P is interpreted as V, where we call D_1 *instantiated* if D_1 is located on destination L.

As shown in the above examples, the `<Prop1>`, `<DTK:Value>`, `<DTK:NewValue>`, and `<DTK:Restriction>` tags are used to define P, V, L, and R respectively.

Restriction R can be any restriction, but the following three types of restrictions would be enough to describe most tickets based on our investigation.

Schema restriction: The schema (or format) of the attached description. It defines properties to be defined and its necessity property, i.e., mandatory or optional.

Property value restriction: The value for the specific property of the attached description. Note that property value restriction can also be defined within the schema of the property if the schema is not assumed to be shared.

Hash value restriction: The hash value of the description. Note that this restriction can only be applied when the attached description has been instantiated when this restriction is specified.

4.3 Composable ticket model

In this section, we propose an approach to implement the (13) composable ticket features. Composability enables the definition of a complex ticket using multiple sub-tickets.

There are many cases when a sub-ticket must be issued separately with the original ticket typically because the tickets are issued by different organizations or issued at different times.

The restriction-specified incomplete/complete link can also be applied to composable tickets. The signed description to be attached can be a ticket.

For example, suppose that a plane ticket can comprise an open ticket and a flight reservation ticket, which are used for a certain air route and reserved for a certain flight on the reserved date, respectively. The reservation ticket can be considered as a sub-ticket of the open ticket and attached to the reservation status property of the open plane ticket.

```
<RDF:RDF>
  <RDF:Description ID="ticket_001" bagID="ticket_bag_001">
    <DTK:Issuer>issuer@airline1.com</DTK:Issuer>
    <DTK:Promise>
      <RDF:Description>
        <MTK:Departure>London</MTK:Departure>
        <MTK:Destination>Boston</MTK:Destination>
        <MTK:Reservation>
          <RDF:Description>
            <DTK:Ticket href="#ticket_002"/>
            <DTK:Restriction>
              <RDF:Description>
                <RDF:InstanceOf href="http://airline1.com/schema#RsvTicket"/>
                <DTK:Issuer>issuer@airline1.com</DTK:Issuer>
              </RDF:Description>
            </DTK:Restriction>
          </RDF:Description>
        </MTK:Reservation>
        <MTK:Conditions>
          <RDF:Description>
            <DTK:Value href="http://airline1.com/conditions"/>
            <DTK:Restriction>
              <RDF:Description>
                <DTK:Digest>476169572bb3680708cd4204907735ac</DTK:Digest>
              </RDF:Description>
            </DTK:Restriction>
          </RDF:Description>
        </MTK:Conditions>
      </RDF:Description>
    </DTK:Promise>
    <DTK:Owner>u1@host1.com</DTK:Owner>
    <DTK:Transferability>NO</DTK:Transferability>
    <DTK:NumberOfTimes>ONCE</DTK:NumberOfTimes>
    <DTK:ValidPeriod>INFINITE</DTK:ValidPeriod>
    <DTK:View href="http://airline1.com/plane_ticket.gif"/>
  </RDF:Description>

  <RDF:Description href="#ticket_bag_001">
    <DTK:Signature>9572bb3680708476735ac16cd4204907</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

Figure 2. Open plane ticket example

Figure 2 shows an example of describing an open plane ticket in the RDF syntax. In this example, the two restrictions of a sub-ticket to be attached are defined.

- The ticket must be an instance of the reservation ticket. This is an example of schema restriction.
- The ticket must be issued by issuer@airline1.com. This is an example of a property value restriction.

The Conditions property in Figure 2 also includes the hash value restriction described in Section 4.2.

The `<RDF:InstanceOf>` and `<DTK:Digest>` tags are used to define schema restrictions and hash value restrictions respectively. To define a property value restriction, the tag for the property, i.e., the `<DTK:Issuer>` tag in this example, is used in the description of `<DTK:Restriction>`.

There are many applications of composable tickets as shown in Table III. It is possible to change the owner property of a transferable ticket by attaching a transfer (ticket) if a restriction-specified link is defined in the owner property. In this case, the restriction is that issuer of the transfer is the transferor.

It is also possible to control anonymity of a ticket by attaching different types of public key (PK) certificates to the owner property of the ticket. That is, if a ticket must be traceable, a PK certificate with user identifier is attached, and if a ticket must be untraceable, a PK certificate without user identifier is attached.

Other examples are a composition of a deferred payment ticket and a check, or an authorized document and the approval stamp as shown in Table III.

4.4 Common processing components

This section explains the prototype system of common ticket processing components that we developed.

Original ticket		Attached ticket / description	
Type	Property	Schema restriction	Value restriction
Any transferable ticket	Owner	Transfer (certificate)	Issuer is the transferor
Any traceable ticket or transfer attached	Owner	PK certificate with user identifier	Issuer is an CA
Any untraceable ticket or transfer attached	Owner	PK certificate without user identifier	Issuer is an CA
Any deferred payment ticket	Payment status	Check or draft	Issuer is a bank
Any documents to be authorized	Approval	Approved stamp	Issuer is the specified issuer
Any ticket details/conditions are described	Conditions	None	Digest value is specified (See Figure 2)

Table III. Application of Composable Tickets

(1) Ticket Editor

The *ticket editor* is a component that generates a ticket skeleton by interacting with the ticket issuer and using pre-defined ticket schemata (layer 1, 2, and 3). A *ticket skeleton* is similar to a ticket except that it may leave some properties blank, e.g., reference number, required tickets, or the signature of the issuer. This is because these property values are usually given dynamically. The GUI of the ticket editor is shown in Figure 3. This component is typically used by ticket designers. Note that defining the ticket schemata is out of scope of this paper.

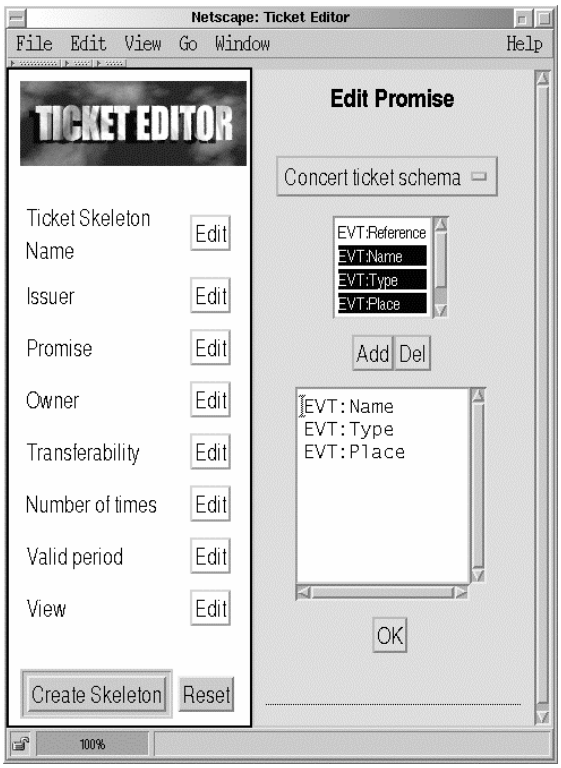

Figure 3. Ticket editor

(2) Ticket Generator

The *ticket generator* is a component that generates ticket instances from a ticket skeleton by interacting with users or customers to whom the ticket is issued. The ticket generator fills in the blank properties by generating or prompting input from the users and signs the ticket. This component is typically used in a Web server application. Figure 4 shows the flow of how the ticket is generated using the ticket editor and generator.

(3) Ticket Reader

The *ticket reader* is a component that checks the validity of the ticket and voids the ticket when a service is rendered. The voiding method depends on the type of the ticket. For example, it can be done by auditing a record in which the ticket owner acknowledges the ticket consumption and signs it. This tool is typically used in ticket examination machines

(4) Ticket Transferor

The *ticket transferor* is a component that changes the ticket owner identity to a new person. Ticket transfers can be done by adding a transfer ticket that specifies the new ticket owner as described in Sections 4.2 and 4.3. This tool is typically used in the transferor's (or ticket owner's) ticket wallet or server.

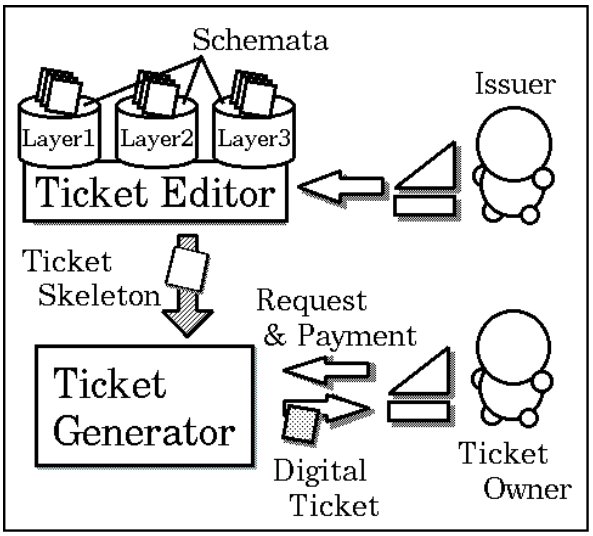

Figure 4. Generating a ticket

(5) Ticket Viewer

The *ticket viewer* is a component that displays the contents (or meaning) of the ticket and checks the validity of the ticket. This component is typically used in the purchaser's ticket wallet and used to confirm the contents of the ticket before conducting a transaction. This reduces the number of problems encountered while trading because ticket purchasers can confirm the contents of the ticket before making a transaction.

Details regarding the processing scheme of these components are out of the scope of this paper. The implementation details will be presented in another paper.

Our current implementation uses an online checking scheme to prevent duplicate redemption. However, more sophisticated protocols [2][7] and/or smart card protocols [14][15] invented for electronic cash can be used if more complete anonymity, transferability, divisibility, and off-line capabilities are required. This paper does not conflict with these protocols, instead one of our goals is independence from the infrastructure for preventing duplicate redemption. It is important to have numerous types of tickets but common components or systems for ticket processing can be shared by establishing parameters for the layer 1 properties as described in this paper.

Additionally, a seller (or reseller) of a ticket can transmit the ticket information to a marketplace without typing it in. The process can be completed by click- or drag & drop-based GUIs. Also, the contents described in the ticket can be read mechanically, thus

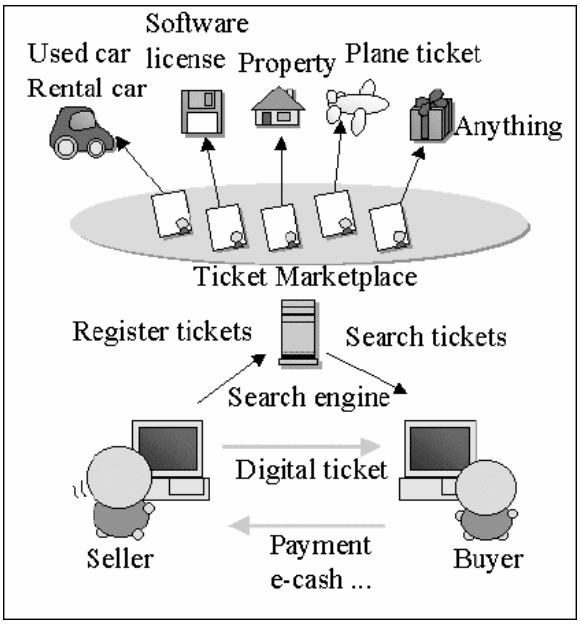

Figure 5. Ticket marketplace

making the purchaser's search for tickets in the marketplace very efficient. This can be expected to promote resale or recycling over the Internet. Figure 5 shows the outline of a ticket marketplace.

5. Conclusions

This paper classified various types of digital tickets and clarified common properties of digital tickets.

A ticket description method was proposed that enables various ticket properties to be defined. The tickets described by the proposed method are machine understandable, state-manageable, and composable. To achieve machine understandability, a ticket using RDF with three-layer ticket schemata was described in Section 4.1. To achieve state-manageability, a new ticket description method that uses restriction-specified incomplete link was proposed in Section 4.2. To achieve composability, a composable ticket model that expresses a composite ticket using a set of required subtickets was proposed in Section 4.3.

Common components that can be used in the issuing, trading, and spending of various types of digital tickets were presented in Section 4.4. These components can be shared among applications because ticket properties of the first layer schema determine the processing patterns regardless of application-specific ticket properties of the second layer schemata defined by each industry.

A common processing platform for digital tickets makes it easy for a small enterprise or individuals to issue or trade their own tickets over the Internet, even though the number of tickets that are issued is small.

There is possibility that all kinds of tickets, including train tickets and admission tickets to amusement parks, will be substituted with digital tickets, and that the ticketing machines will be substituted with Web terminals when smart cards [18] become more popular. To make it practical, a standard ticket description method and ticket processing infrastructure must be established and we believe that our framework presented here is the key.

Acknowledgements

We would like to thank Yuji Takehisa, Jun Sekine, Yasunao Mizuno, Kazuo Matsuyama, Masayuki Terada, Hiroshi Kuno, Ayumu Eto, and the referees for their suggestions.

References

[1] N. Asokan, P. A. Janson, Michael Steiner, and M. Waidner, "The State of the Art in Electronic Payment Systems," *IEEE Computer*, Sep. 1997, pp. 28-35.

[2] D. Chaum, "Privacy Protected Payments Unconditional Payer and/or Payee Untraceability," In *Smart Card 2000*, North-Holland, Amsterdam, 1989, pp. 69-93.

[3] G. Davida, Y. Frankel, Y. Tsiounis, and M. Yung, "Anonymity Control in E-Cash Systems," In *Proceedings of Financial Cryptography '97*, LNCS 1318, pp. 1-16.

[4] Eran Gabber, Phillip B. Gibbons, Yossi Matias, and Alain Mayer, "How to Make Personalized Web Browsing Simple, Secure, and Anonymous," In *Proceedings of Financial Cryptography '97*, LNCS 1318, pp. 17-31.

[5] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro, "The Millicent Protocol for Inexpensive Electronic Commerce," In *Proceedings of WWW4*, http://www.w3.org/Conferences/WWW4/Papers/246/

[6] J. W. Matonis, "Monetary Freedom," In *Proceedings of INET95*, Internet Society, http://info.isoc.org/HMP/PAPER/136/html/paper.html

[7] T. Okamoto and K. Ohta, "Universal Electronic Cash," In *Advances in Cryptology, Proceedings of CRYPTO '91*, J. Feigenbaum (Ed.), LNCS 576, pp. 324-337.

[8] Peter Wayner, "Digital Cash," Academic Press Ltd., 1997.

[9] Extensible Markup Language (XML) specifications, The World Wide Web Consortium, 1998, http://www.w3.org/XML/

[10] Resource Description Framework (RDF) Model and Syntax, The World Wide Web Consortium, Working Draft, 1998, http://www.w3.org/TR/1998/WD-rdf-syntax-19980216

[11] Resource Description Framework (RDF) Schemas, The World Wide Web Consortium, Working Draft, 1998, http://www.w3.org/TR/WD-rdf-schema

[12] E-Stamp Corporation, "E-Stamp," http://www.e-stamp.com/

[13] Gold & Silver Reserve, Inc., "e-gold," http://www.e-gold.com/

[14] Java Card Forum, http://www.javacardforum.org/

[15] MONDEX International Ltd., http://www.mondex.com/

[16] The NetBill Electronic Commerce Project, http://www.ini.cmu.edu/netbill

[17] Secure Electronic Transaction (SET) specifications, http://www.mastercard.com/set/

[18] Smart Card Forum, http://www.smartcrd.com/

[19] Transaction Net, "Complementary Currencies and Exchange Networks," http://www.transaction.net/money/comp/