Verifying Identifier-Authenticity in Ubiquitous Computing Environment

Tetsuo Kamina[†,‡] Toshinori Aoki[‡] Yoshiteru Eto[‡] Noboru Koshizuka[†,‡]
Jun Yamada[‡] Ken Sakamura[†,‡]

†The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
‡YRP Ubiquitous Networking Laboratory
No.28 Kowa Building, 2-20-1 Nishigotanda, Shinagawa-ku, Tokyo 141-0031, Japan

Abstract

In ubiquitous computing environment, identification of objects and places in the real world is important, and 2-D printing code is useful to store identifiers of them. However, since it is easy to modify the content stored in the 2-D code, we must verify whether the identifier written in the 2-D code is indeed issued by the authorized organization. In this paper, we propose a verification method for identifiers stored in the 2-D code. Our system makes sure that the identifier is indeed issued by the authorized organization, and the size of pair of identifier and its authenticator is small enough to be written in a 2-D code. Furthermore, the proposed system has compatibility with the existing 2-D code reading software. Our system is now operated in practical use, and some services based on this mechanism have been developed.

Keywords: 2-D code, QR Code, infrastructure, authenticator, ucode.

1. Introduction

In ubiquitous computing environment, it is important to enable mobile computing terminals to *identify* objects and places in the real world. There are many methods to achieve this requirement[9] and one of the most promising ways for it is to attach a *tag* containing identifier to the object or place and provide a tag reader to the mobile terminal.

There are many devices to implement such tags. For example, we may use RFID for this purpose; however, the most costless and easiest way to implement tags is using *printed tags* such as barcode and 2-D printing code. For example, QR Code[5], which is one kind of 2-D code widely used in Japan, can contain character strings including digits; therefore, such a tag is useful to store identifiers represented in digits or strings.

We have used QR Code tags in many ubiquitous computing experiments and demonstrations; e.g., in an experiment on food distribution tracing, we used QR Code tags to identify each foodstuff to trace information of it, to provide detailed information for consumers (Since the amount of information that can be written in a QR Code tag is limited, only the identifier is written in it. Other information is stored in the online servers that we call *information servers*). Furthermore, to recognize places in the real world, we have attached a large amount of QR Code tags to identify each spot (Figure 1). In this experiment, we developed a navigation system that provides not only geographic information (e.g. latitude and longitude) but also *semantic information* about that spot such as the name of building, how to go up to the conference room in that building, and so on. Such information is especially useful for visitors, unless we can properly identify each spot.

However, there has been a problem. Since it is easy to forge a 2-D code such as QR Code, and each tag is attached in the public space, someone can maliciously overwrite the tag, which actually happened in our experiences. Since identification is so important in ubiquitous computing, we must make sure that the identifier stored in the 2-D code is indeed issued by an authorized organization.

In this paper, we propose a verification method for identifiers stored in the 2-D code. Basic idea of the proposed method is that additional information indicating the validity of identifier, which we call *authenticator* of the identifier, is stored into the tag in addition to the identifier. There are many technologies that are possibly useful to implement the idea; among them, we have deliberately investigated which one is most appropriate for our purpose and concluded that using a keyed hash function is one of the most reasonable ones.

Based on this idea, we have implemented the verification

Figure 1. Printed tag attached to one spot of Shinjuku, a famous town in Tokyo

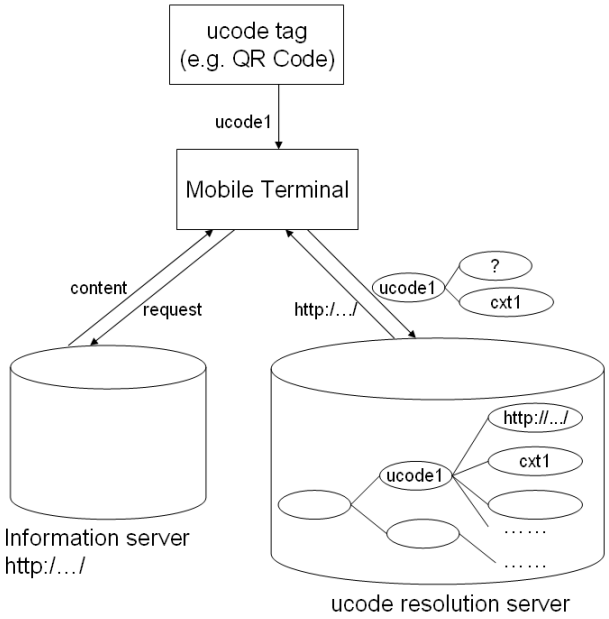

Figure 2. Ubiquitous computing architecture we based [1]

server that maintains the secret key for keyed hash function and verifies the authenticators. The proposed system makes sure that the identifier stored in a 2-D code is indeed issued by the authorized organization, and the pair of identifier and its authenticator is small enough to be written in a QR Code tag. Furthermore, the proposed system is compatible with the existing QR Code reading applications that are normally implemented on cell phones in Japan. The verification server is now operated in practical use. Some services based on this mechanism have been developed, including *network-linked magazines* that bind online contents (which can be updated at any time) with the printed articles. Furthermore, we are undergoing an ubiquitous computing experiment in which we have attached QR Code tags to lampposts and other spots (such as shops, banks, etc.) in Ginza, one of the most famous town in Tokyo. Based on the proposed system, these QR Code tags securely contain identifiers of each spot, and many information services are provided.

The rest of this paper is structured as follows. In section 2, we introduce the ubiquitous computing architecture we based as background information, and list the requirements of the proposed system. Section 3 presents some candidate technologies to implement the requirements, and discuss which is the most reasonable one. Section 4 shows an implementation of the proposed system and an example service based on it. In section 5, we show some related work. Finally, section 6 concludes this paper.

2. Requirements for the Proposed System

2.1. Background

Before we proceed, we briefly introduce the ubiquitous computing architecture we based.

Figure 2 summarizes the architecture. In ubiquitous computing environment, it is important that every object and place in the real world is identified by mobile terminals. In our architecture, this requirement is achieved by attaching a tag containing identifier of the object or place to be read by using tag readers installed on the mobile terminals. The identifier, which we call *ucode* (ubiquitous code), has 128 bit length, which has enough size to identify every object in which we are interested.

In our architecture, many kinds of tags may be used to store ucodes. For example, passive RFID, active RFID in which a battery is built so that it can send radio wave by itself, and infrared markers may be used. Of all others, using 2-D printed code is one of the most useful ways to store ucode, because it is easy to create and there already exists

many kinds of 2-D code readers (e.g., readers for QR Code, a kind of 2-D code, are normally installed on cell phones in Japan).

Since the ubiquitous computing environment should be a general purpose infrastructure, the tags should *not* contain any application specific information; instead, the bindings between ucodes and application specific data are maintained in the online servers that we call *ucode resolution servers*. The ucode resolution servers are maintained by the authorized organizations that manage the ucodes. The mobile terminal can connect to the ucode resolution server to query which information is bound to the ucode written in the tag. Note that such information may be an address of the *information server* that actually maintains the application specific data.

For example, in the application where a mobile terminal reads a ucode tag that is attached in front of a restaurant and then displays today's recommended dinner menu, the process proceeds as follows; at first, the mobile terminal reads a ucode from the tag and send a query to the ucode resolution server to fetch the information bound to the ucode; second, the ucode resolution server responds to it with the location (e.g. URL of the restaurant) where the information will be found; finally, the mobile terminal sends a request to the information server where the URL points and get the information about the dinner menu.

2.2. Problems and Requirements

As mentioned in the previous subsection, we consider that our architecture should be a general purpose ubiquitous computing infrastructure where many special purpose applications are constructed. In our architecture, numerous ucode tags are distributed in the public spaces to identify objects and places. The problem is that creating a 2-D code is so easy that it is very probable that someone maliciously overwrite 2-D code placed in the public infrastructure. Therefore, the authorized organization should be able to verify that the ucode written in the 2-D code is indeed issued by the authorized organization.

The problem may be solved by storing additional information indicating that the ucode is issued by an authorized organization in the tag, in addition to the ucode. However, we should take into account that most 2-D codes provide only limited space to encode information, thus the widely used digital signatures are hard to be stored in them as authenticators. For example, in the RSA based digital signature algorithm[10], it is said that the length of signature should be at least 2048 bits for ensuring the safety. Therefore, we have to choose a method in which authenticators may be small enough to be encoded in 2-D code, without loss of security.

There still exists some problems. For example, after reading the tag, the user can acquire the pair of ucode and its authenticator; the user then can print and put it elsewhere to deceive other users. We call it copy attack. Such copy attacks may be detected in some extent. For example, for ucodes that identify spots in public spaces, we can bind a ucode with geographic information. If a ucode is copied to another place, a contradiction will be found so that the copying will be detected. Another problem arises when some malicious users legally acquire the authority to issue ucodes. In such a case, we consider that the safety of the system should be discussed not only from technological points of view but also from legal and sociological points of view. This paper, however, does not aim to reveal these issues. Instead, this paper aims to reveal the basic mechanism of ensuring that ucodes are indeed issued by the authorized organizations.

Finally, since 2-D code readers and software are normally installed on many mobile devices, we consider that the 2-D code tag used in our system should also be able to processed by using these software; i.e., our 2-D code tag should be compatible with the existing software.

All in all, we summarize the requirements that the proposed system should satisfy:

- The proposed system must be able to make sure that the ucode is indeed issued by the authorized organization.
- The authenticator must be small enough to be written in 2-D code, without loss of security.
- The proposed system should be compatible with the existing 2-D code readers and software.

3. Technical Elements

There are many technologies that are possibly useful to implement the aforementioned requirements. In this section, we discuss which technology is the most reasonable one among them.

3.1. Digital Signatures

To implement the first requirement, one may consider that we can store a digital signature into the tag in addition to the content to verify the validity of identifier. However, as mentioned in the previous section, the widely used digital signature schemes[10, 8] require a large amount of data to store a signature, which conflicts with the second requirement. One solution to this problem may be using the Elliptic Curve Cryptography (ECC)[7], which is considered to be useful to reduce the size of signatures. These algorithms

have been proved to be safe and become popular, thus using digital signatures seems to be feasible if we use such algorithms.

If we use this approach, the verification of authenticators will be performed on the mobile terminals. Only the mobile terminals in which signature verification mechanism is installed can be used for this approach.

3.2. Keyed Hash Functions

There is another way to guarantee that the identifier is not modified: calculating the *hash* of identifier using a keyed hash function. To make the keyed hash function secure, it is also required that the size of hash should not be small; however, the size of hash does not have to be very large. For example, we may use the SHA-1 algorithm[2], whose hash size (160 bits) is small enough to store hash into the QR Code tag in addition to the ucode. The problem is how to manage the secret keys used to calculate hashes. Management of keys should be centralized on a secure server.

If we use this approach, the verification of authenticators cannot be performed on the mobile terminals. Instead, we have to install a verification server that manages the secret keys for hash functions and performs verification of authenticators.

3.3. 2-D Code and Encoding

There are many kinds of 2-D code systems[5, 3, 4]; among them, we chose QR Code[5] for encoding ucode, because it is useful to store a serialized ucode represented in characters and digits strings, and it is widely used in Japan, thus QR Code readers are normally installed on cell phones.

The mobile terminal explained in Figure 2 communicates with the ucode resolution server using special purpose software and special purpose protocols. On the other hand, QR Code readers installed in cell phones interpret URLs written in the QR Code and communicate with servers using HTTP. To ensure compatibility to existing software, a mapping from special purpose protocols we have developed to HTTP should be considered.

4. Design and Implementation

Based on the previous discussion, we have implemented a security enhancement on our architecture. The overall description is summarized in the data flow shown in Figure 3 and sequence diagram shown in Figure 4.

4.1. Issuing and Signing

The proposed system supports both of public-key cryptography based digital signatures and keyed hash functions.

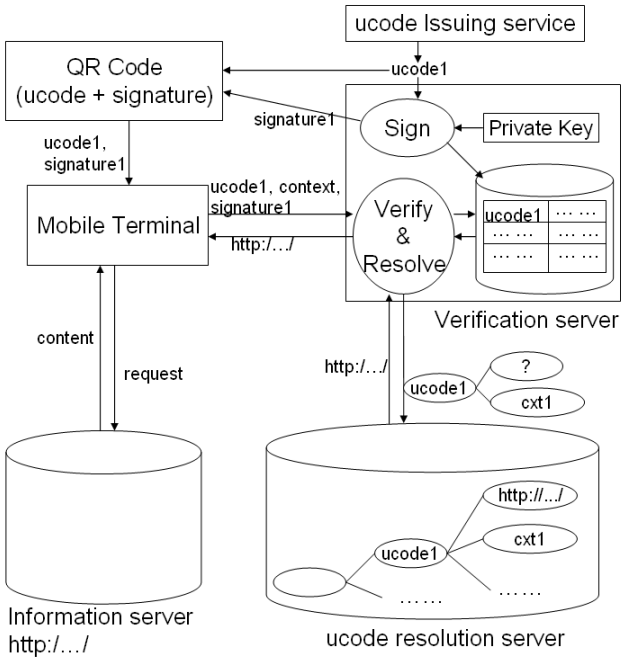

Figure 3. Data flow of security enhancement on our architecture

To provide compatibility to the existing software, or to support keyed hash functions, we implemented a verification server that is maintained by an authorized organization.

At first, a person who wants to attach a ucode tag to an object or place (we call this person *owner*, because she is possibly the owner of the object or place) acquires a user account of the ucode issuing service, and requests a ucode using this account. The ucode issuing service then issues a ucode and send a request to the verification server to sign the ucode. The key used for signing is secretly maintained in the verification server, thus signing is performed on the verification server. The verification server signs the ucode, stores the pair of ucode and its authenticator into the database, and returns the authenticator to the ucode issuing server. The ucode issuing server then creates a QR Code tag (a binary bitmap image) and returns it to the owner (for example, by displaying it on the Web browser).

The owner can select which algorithm, public-key cryptography or keyed hash function including MD5, SHA-1, SHA-256, and so on, is used to sign the ucode.

4.2. Building and Using Applications

The owner prints the QR Code tag and attaches it on the object or place. Based on the ucode tags distributed in many objects and places, we can construct many kinds of applica-

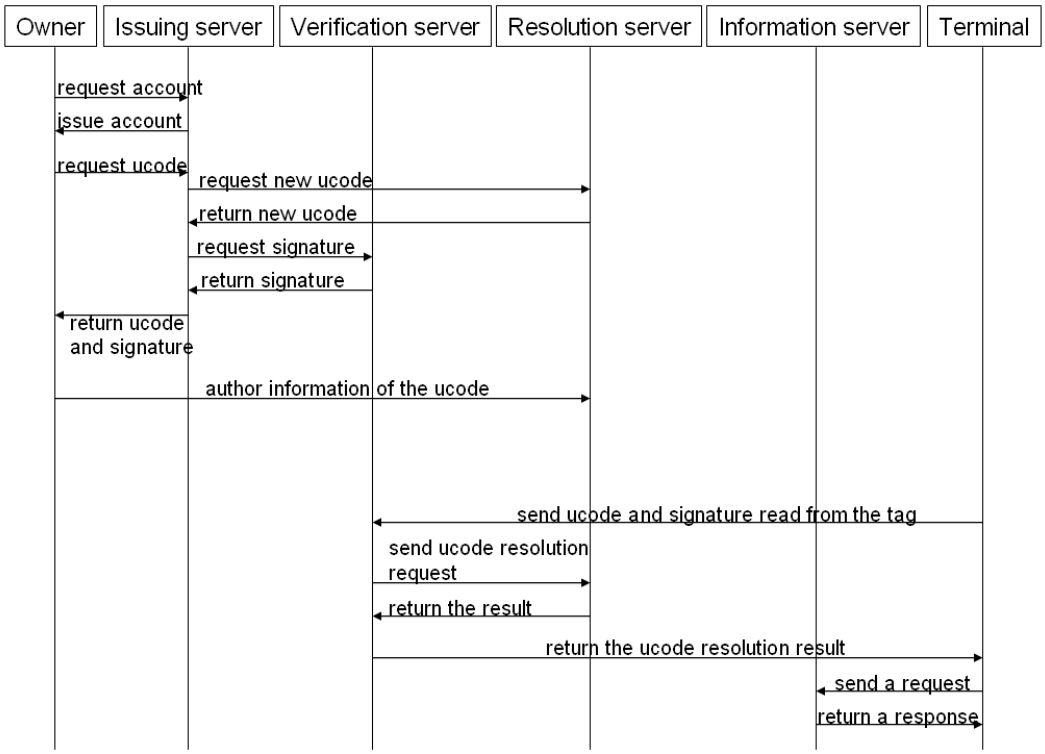

Figure 4. Sequence diagram of the proposed system

tion services. To build a service, the application developer (possibly but not necessarily the same as the owner of the tags) registers information related to the ucode to the ucode resolution server.

The user of the application service acquires the information that the application developer has registered by reading ucode and its authenticator from the tag. For this purpose, many kinds of mobile terminals may be used; for example, we can use our original, special purpose software. However, to encode the pair of ucode and authenticator, we should take into account the compatibility to the existing software such as QR Code readers on cell phones that interpret URLs encoded in the QR Code and get the contents from the URL.

If the user uses the special purpose software, the process is straightforward; the mobile terminal accesses the verification server to verify the ucode and its authenticator. After the verification, the verification server sends a query to the ucode resolution server on behalf of the mobile terminal to get the information bound to the ucode and returns the result to the terminal

The existing software, however, does not know where is the verification server. We therefore encode the ucode and its authenticator in the form of URL as follows:

```
http://<verification server>/<path name>
?X-UIDC-UCODE=<ucode>&X-UIDC-SIGNATURE=
<authenticator>&X-UIDC-ALGORITHM=
<algorithm>
```

where `<ucode>` and `<authenticator>` are represented in hex digits.

The legacy software interprets the URL and sends an HTTP request to the verification server. To connect to the legacy software, a Web service is running on the verification server. Ucode, its authenticator, and algorithm that is used to calculate authenticator (and possibly some application specific context information) are augmented using HTTP request parameters. The verification server then performs ucode resolution as in the case of communicating with the special purpose software, and returns the result in the form of HTTP response. Therefore, the legacy software can be reused without any modification.

Note that this encoding is only for assuring the compatibility to the existing software. There is no guarantee that the URL written in the QR Code tag is valid one. Therefore, it is recommended to use the special purpose software where the list of reliable verification servers are installed.

4.3. Running Systems

The proposed system is now operated in practical use. Based on this system, a *network-linked magazine* is

published[11]. This service binds an article on printed magazine with an online content that provides detailed and up-to-date information about that article. On each article, a ucode is issued and a QR Code tag containing the ucode and its authenticator is attached. Readers of the article can get more detailed information about that article by reading the QR Code tag by using their own QR Code readers. This service reduces amount of pages of printed magazines thus reducing the cost for publishing the magazine. Furthermore, the online article can be updated realtime, thus readers are provided up-to-date information. Ucodes are signed and verified by using the proposed system.

We are undergoing an experiment in which we have attached QR Code tags to lampposts and other spots (such as shops, banks, etc.) in Ginza, one of the most famous town in Tokyo. For example, QR Code tags attached to lampposts are used to identify the place to show the information around the place, such as list of shops nearby the place. These QR Code tags are securely signed and verified by using the proposed system.

Furthermore, Nihon Unisys clearly stated that they use our system in their equipment management. These case studies will provide more insight in this research field.

5. Related Work

There exists some related researches on signing contents written in 2-D code. Especially for QR Code, Ito et al. proposed an authentication method using 3-D pattern communication to use QR Code in authentication[6]. Using the display of cell phones, this method displays a number of codes one after another thus using 3-D pattern to encode information that is too large to be stored into the 2-D QR Code. Based on this method, they built an e-ticket issuing system as a case study[13]. However, this method cannot be used for the printing tags that have no displaying devices.

Toye et al. proposed a mobile service toolkit (MST) that enables mobile devices such as cell phones to access site-specific services [12]. As in our system, MST provides a way to connect the online server that provides site-specific services by reading *visual tags* similar to QR Code tags. However, the system's architecture is less generic; in this system, address of the server and application specific data are written in the tag, while in our system, a tag contains only an identifier.

6. Concluding Remarks

This paper presents how to verify identification in the ubiquitous computing environment. By attaching additional information (authenticator) that indicates the validity of ucode, our system makes it possible to guarantee that the ucode is indeed issued by the authorized organization. To generate authenticator, we choose a method using a keyed hash function to calculate a hash of the ucode among some candidate technologies. Verification is securely performed on the verification server. The size of pair of ucode and its authenticator is small enough to be written in the QR Code tag. Furthermore, the proposed system has compatibility to the existing QR Code reading applications implemented in vast series of cell phones.

Acknowledgments. We thank infrastructure research group and secure networking research group in YRP Ubiquitous Networking Laboratory for fruitful discussion. Especially, Shingo Miyazaki, Katsunori Shindo, and Chiaki Ishikawa gave us very helpful comments on improving this paper. The experiment in Ginza is sponsored by Tokyo Metropolitan Government and Ministry of Land, Infrastructure and Transport Japan.

References

[1] Ubiquitous ID Center. http://www.uidcenter.org/.

[2] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174, 2001.

[3] ISO/IEC15438:2001. Information technology – automatic identification and data capture techniques – bar code symbology specifications – PDF417.

[4] ISO/IEC16023:2000. Information technology – international symbology specification – MaxiCode.

[5] ISO/IEC18004:2000. Information technology – automatic identification and data capture techniques – bar code symbology – QR Code.

[6] M. Ito, R. Uda, K. Awaya, S. Nasu, G. Inomae, H. Shigeno, K. Okada, and Y. Matsushita. Authentication with 3-D pattern communication. *Trans. IEICE*, J86-B(4):619–629, 2003. (In Japanese).

[7] N. Koblitz. Algebraic aspects of cryptography. *Algorithms and Computation in Mathematics*, 3:148–178, 1998.

[8] National Institute for Standards and Technology. The digital signature standard. *Communications of the ACM*, 35(7):36–40, 1992.

[9] P. Peursum, H. H. Bui, S. Venkatesh, and G. A. West. Using interaction signatures to find and label chairs and floors. *Pervasive Computing*, 3(4):58–65, 2004.

[10] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[11] K. Sakamura, editor. *TRONWARE*. Number 101. Personal Media, 2006.

[12] E. Toye, R. Sharp, A. Madhavapeddy, and D. Scott. Using smart phones to access site-specific services. *Pervasive Computing*, 4(2):60–66, 2005.

[13] R. Uda, S. Ichimura, and M. Ito. E-ticket issuing system with 3-D pattern recognition for mobile terminals. Collected papers on Mito Software Projects, 2003. (In Japanese).