

# Implementation of Secondary Available Digital Content Protection Schemes using Identity-based Signatures

Nozomi Nagashima<sup>1</sup>, Masaki Inamura<sup>2</sup><sup>a</sup> and Keiichi Iwamura<sup>1</sup>

<sup>1</sup>Graduate School of Engineering, Tokyo University of Science, Tokyo, Japan

<sup>2</sup>Center for Research and Collaboration, Tokyo Denki University, Tokyo, Japan

**Keywords:** Copyright Protection, Edit Control, Digital Signature, ID-based Signature.

**Abstract:** User generated content (UGC) is widespread, wherein general consumers utilize the Internet to generate content files. Using this service, content files can be easily created and uploaded. However, copyright is often not protected on the Internet. An activity known as a creative commons license enables authors to manage the secondary use of original content files. However, this license cannot prevent malicious editors from using the original files. Therefore, we propose technical protection measures using the creative commons license and identity-based signature, wherein authors apply identity-based signatures to content files that need to be protected. By verifying the signature, malicious edits performed on the files can be detected. We investigated the processing speed required for this via simulation and used the 3DCG movie editing tool "Miku Miku Dance" as the content files.

## 1 INTRODUCTION

The development of the Internet has enabled the easy generation and publishing of content files. In particular, services such as YouTube and Nico Nico Douga allow users to easily publish content files. User-generated content (UGC) enables browsing and retrieving such files published by other users. Therefore, the distribution of secondary content files has become popular. Herein, it is necessary to protect the rights of the developer of the original content files when used as secondary content files. The Creative Commons license enables the author to manage the secondary use of his original content files. However, editors can freely adhere to a Creative Commons license. Consequently, malicious editors cannot be prevented from making malicious edits to these files. Therefore, to prevent the aforementioned issues, we herein propose technical protection measures developed by combining the Creative Commons license and ID-based signature.

We implemented these copyright protection technologies using a digital signature. The target content files are a 3DCG model developed using the 3D video creation tool "Miku Miku Dance". On UGC, there are several advancements that combine studies


performed from multiple creators. Therefore, assuming multiple authors, we adopted ID-based signatures that do not require verification of public key certificates.

In the proposed method, the secondary user can only edit with the author's permission. Verification fails if an edit is performed without the author's permission. If the secondary user complies with the author's intentions, he or she could use the original content files without infringing on the author's rights. This is achieved by disabling the unauthorized use of the original content files.

Chapter 2 presents an overview of the existing literature. Chapter 3 presents the entity, and Chapter 4 presents the algorithm. The simulation results are discussed in Chapter 5, and the conclusion is presented in Chapter 6.

## 2 RELATED WORK

In section 2. 1, we describe existing research on a scheme that is used for managing content editing. Tanaka et al. partitioned one 3D model into multiple content files, thereby enabling the editing of each

 <https://orcid.org/0000-0001-8056-3820>

body component (Tanaka et al, 2017). Thereafter, the signature method described in Section 2. 2 was used for protection. In the scheme developed by Tanaka et al., editing of multiple content files was managed using the Boneh Lynn Shacham signature (Boneh et al, 2003). Furthermore, Inamura et al. proposed an order-specified multisignature scheme capable of investigating the order in which the signers sign a document (Inamura et al, 2011).

## 2.1 Content Editing Method

Tanaka et al. considered content files of multiple 3D models as humanoid characters. "Miku Miku Dance" (Higuchi, 2020) enables the creation of dance videos through the movement of a humanoid character; in this tool, the coordinates of the hands and feet of the 3D model are set for each movement. Tanaka et al. partitioned this 3D model into multiple content files such that each part of the body, such as the right arm and the left foot, allows detailed editing management for each part of the body. Permitted editing methods include deleting, changing, and adding another content file. However, the inheritance function that ensures a secondary user adheres to the editing adjustment specified by the author is not implemented.

### 2.1.1 Entity

Following the scheme developed in the previously mentioned study, an *i*-th author was introduced and the word "editor" was not used. Therefore, the two entities called the *i*-th author and verifier are described as follows:

#### 1. The *i*-th author

The *i*-th author is involved in the operations and can set the content file to edit the control signature and update the aggregate signature. The operation of this author can be represented with the help of a tree structure, such as the one depicted in Fig. 1. In the scheme proposed by Tanaka et al., the authors located in the root part of the tree are called the first author. If the height of the tree is *n*-1, then the section of the tree root is called the *n*-th author. Therefore, *i* is defined as the author's position during the operation. The *i*-th author can set an edit control signature on the developed content file. We can edit if the edit control signature defined by the *i*-1st author permits editing. A11 through A16 in Fig. 1 depicts the principal content developed by at least two first authors. The second author used the primary content files of the first author to

develop the secondary contents, A21 and A22. Finally, the third author developed the final content A31. Consequently, the second author edited the content file when the edit control signature settings set by each first author permitted editing. The third author adhered to the edit control signature settings set by the first and second authors.

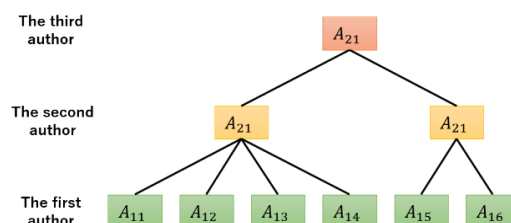


Figure 1: Entities in the tree structure of content files.

#### 2. Verifier

The verifier verifies if the content file has a valid signature. By enabling verification on the device in which the content is played, a system can be developed that cannot play the content without a valid signature.

### 2.1.2 Edit Control

First, the operation of the creator of the content file is explained. The creator of the content file can set whether or not to allow content editing. Editability is converted into parameters and signed together with the content file. There are two types of parameters, "No Derivative" and "ShareAlike". If the creator of the original content permits editing, the editor that secondarily uses the content file can change the work. For example, one can change the color of the work or move the character's arms and legs to change the pose. If the " No Derivative " content file is changed, it is judged to be invalid. Next, " ShareAlike " is described. Set "ShareAlike " when you want the editor to comply with the settings set by the creator of the original content. For content files for which " ShareAlike " is set, the editor after secondary use cannot change the editing control set by the creator. This protects the creator's willingness not to change the work.

Next, the editor that secondarily uses the content file is described. The editor verifies the signature of the original content. If editing is performed without observing the editing control set by the creator of the content file, it is judged as invalid. If the creator of the original content has not set " ShareAlike ", the editor can set new editing controls.

## 2.2 Digital Signature Scheme

### 2.2.1 BLS Aggregate Signature

Tanaka et al. used the aggregated signature by BLS. Boneh, Lynn and Shacham proposed an aggregate signature scheme based on the BLS signature scheme using the operation on an elliptic curve and pairing. This scheme aggregates at least two signatures for every message into one signature with a steady length independent of the number of signers.

denote the set of signer groups and the set of signer symbols that participate in generating an aggregate signature, respectively. The construction of the aggregate signature is as follows.

**Key Generation:**  $g$  is a generator of  $\mathbb{G}_1$ .  $x_i$  is a value of  $Z_p$ . The key generation center is calculated  $v_i = x_i g$ . Herein,  $x_i$  is a private key of  $u_i \in L$ , and  $v_i$  is a public key of  $u_i$ .

**Signing:** Let  $H: \{0,1\}^* \rightarrow \mathbb{G}_2$  be a one-way hash function. Let  $m_j$  be a signer's message  $u_j$ . Then, the signer  $u_j$  calculates  $h_j = H(m_j)$  and sets  $\sigma_j = x_j h_j$  as the signature corresponding to  $m_j$ . After signing, we retrieve all signatures and calculate an aggregate signature  $\sigma = \sum \sigma_j (j \in J)$ .

**Verification:** The verifier retrieves  $m_{i_1}, \dots, m_{i_t}, \sigma, g$  and verification keys  $v_j (j \in J)$ . Thereafter, it calculates  $h_j = H(m_j)$  from  $m_j$ s and determines if the following is achieved using pairing.

$$e(g, \sigma) = \prod e(v_j, h_j) (j \in J) \quad (1)$$

If the aggregate signature is accurately developed, the upper equation is achieved.

### 2.2.2 ID Based Aggregate Signature

Unlike BLS signatures, which generate keys from random numbers, ID-based signatures generate keys from author IDs, which represent the author information set for each author. This eliminates the need to issue a public key certificate for signature verification. It utilizes a pairing function with bilinearity characteristics on elliptic curves. Next, the ID-based aggregate signature algorithm is used. The ID-based aggregated signature can aggregate multiple generated signatures into one.

Let  $\mathcal{U} = \{u_{i_1}, \dots, u_{i_t}\}$  be the set of signers participating in the aggregate signature. Let  $\mathcal{J} = \{i_1, \dots, i_t\}$  be the set of participants' symbols in the aggregate signature.

**Preparation:** From the generators  $g \in \mathbb{G}_1$  and  $s \in Z_p$ , the secret key issuing center calculates  $g_{pub} = sg$ , and  $s$  is the master secret key.

**Key Generation:** Let the ID information of the signer  $u_i$  be  $ID_i$ . The private key issuance center uses the one-way hash function  $H_1: \{0,1\}^* \rightarrow \mathbb{G}_2$  to calculate  $Q_{ID_i} = H_1(ID_i)$ . Issue  $d_{ID_i} = sQ_{ID_i}$  to the signer as the private key  $u_i \in \mathcal{U}$ .

**Signing:** Let  $m_i$  be the plaintext signed by the signer  $u_i$ . The signer generates a random number  $r_i \in Z_p$  and is calculated  $U_i = r_i g$ . Then,  $h_i = H_2(ID_i, m_i, U_i)$  is calculated using the unidirectional hash function  $H_2: \{0,1\}^* \rightarrow \mathbb{G}_2$ . Then generate  $V_i = d_{ID_i} + r_i h_i$ . Let  $\sigma_i = \langle U_i, V_i \rangle$  be the signature for  $m_i$ .

**Aggregation:** The signature aggregator collects  $V = \sum V_i (i \in \mathcal{J})$  all the signers  $u_i (\in \mathcal{U})$  and calculates  $V_i$ . Let the aggregate signature be  $\sigma = \langle U_{i_1}, \dots, U_{i_t}, V \rangle$ .

**Verification:** The verifier collects  $g, g_{pub}, U_{i_1}, \dots, U_{i_t}, V, m_{i_1}, \dots, m_{i_t}, ID_{i_1}, \dots, ID_{i_t}$  and calculates  $Q_{ID_i} = H_1(ID_i)$  and  $h_i = H_2(ID_i, m_i, U_i)$ . Determine if  $e(g, V) = \prod e(g_{pub}, Q_{ID_i}) e(U_i, h_i) (i \in \mathcal{J})$  holds.

## 3 USE OF EXISTING SYSTEM

In this section we explain the existing technology used in the proposed method. Initially, the "Miku Miku Dance" is described in Section 3. 1. This software was used as a content file. The creative commons license is described in Section 3.2. We utilize the findings of previous studies on the secondary use of files under a creative commons license.

### 3.1 Miku Miku Dance

"Miku Miku Dance" is a 3D music video production tool distributed freely by Yu Higuchi. Using this tool, music videos can be developed by operating the character "Hatsune Miku." Several videos have been developed using this tool. I have developed such dance movies, but only using humanoid characters such as Hatsune Miku. However, recently, four-legged animals and other humanoids are also available. In addition, props, stages, and backgrounds perfect for music videos have been developed. Several useful materials for developing these videos are available for free. The "Miku Miku Dance" is widely used because music videos can be easily developed using this tool; furthermore, the tool is very user friendly, in particular, for beginners. Fig. 2 depicts an operation screen of the "Miku Miku Dance".



Figure 2: Operating screen of “Miku Miku Dance”.

The character at the center of the screen was Hatsune Miku. Here, the 3D model of this character is used as the content file. However, many 3D models other than humanoid characters are now emerging. In a non-human 3D model, it is not necessary to divide the content file for each part of the human body, as in the previous research. Therefore, in this study, the 3D model itself is regarded as a single content file.

### 3.2 Creative Commons License

The copyright protection system proposed uses a creative commons license for editing management. A creative commons license is proposed as a new copyright rule for the Internet age. Using this license, the author can express his/her intention to “using the work freely if certain conditions are met” However, we cannot protect the operation from malicious secondary users. Therefore, in this study, electronic signatures are combined to protect the creative commons license technically. For example, a malicious secondary user can modify a task that is not allowed to be modified. Herein, if a signature is attached to the task, it is revealed through verification that editing is not permitted. This protects the rights of the author. Thereafter, the types of editing permission are described. The creative commons license has four types of conditions: Attribution, Noncommercial, No Derivative, and ShareAlike. By displaying the icon shown in Fig. 3, authors indicate their intention on the secondary use of the work. The conditions are described as follows:

**Attribution:** This license enables others to adapt and advance an existing task with relevant acknowledgement of the original creator.

**Noncommercial:** A task cannot be used commercially.

**No Derivative:** A task cannot be shared with others in adapted form.

**ShareAlike:** All new works based on the task will use the same license.

In this program, you can set No Derivative and ShareAlike among these licenses.

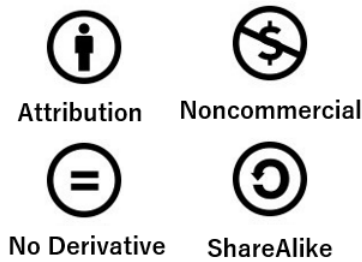


Figure 3: Creative Commons License icon.

## 4 ALGORITHM

We describe the algorithm that generates ID-based signatures and protects content files using the scheme proposed by Tanaka et al (Tanaka et al, 2017). We assume that the binding between the signers and the verification keys is guaranteed by a certification authority and that the information prepared does not come from a third party.

**Key Generation:**  $ID_{ij}$  is the author ID defined based on the location of a task. For example,  $ID_{11}$  denotes the author of the contents  $A_{11}$  in Fig. 1. The key generation center calculates the following:

$$g_{pub} = sg \tag{2}$$

from the generator  $g \in \mathbb{G}_1$  and the master secret key  $s$ . Thereafter, the key generation center calculates the public key and the secret key of the signer  $ID_{ij}$  by

$$Q_{ID_i} = H_1(ID_i) \tag{3}$$

$$d_{ij} = sQ_{ij} \tag{4}$$

Here,  $Q_{ij}$  denotes the public key, and  $d_{ij}$  denotes the secret key. The signing keys display differences.

**Signing:** The author performs the signature generation process prior to publishing the original content that has obtained an administration signature. To prevent duplication, the content ID of each content is different, and the author is identified from the content ID (where the first half of  $IC_{ij}$  is equal to  $ID_{ij}$ ).

#### 1. Setting ID

Author  $ID_{ij}$  determines the control permissions of change, deletion, addition, diversion, and succession for each partial content and the composition of the content. Author  $ID_{ij}$  determines the content ID, where  $IC_{ij}$  is set to zero for partial contents that can be diverted.

#### 2. Determining parameters

Author  $ID_{ij}$  uses the parameters  $p_c$  and  $p_s$  to indicate whether it can be modified and inherited to generate a hash value and an edit management signature. The parameters are  $p_c =$



1 if alterations are allowed and  $p_c = 0$  otherwise. Similarly, the parameter is  $p_s = 1$  if inheritance is set and  $p_s = 0$  otherwise.

$$h_{ijk} = H(IC_{ij} \parallel I_{ijk} \parallel H(A_{ijk}^*) \parallel p_s \parallel p_c \parallel r) \quad (5)$$

$$\omega_{ijk} = r_{ijk} h_{ijk} + d_{ij}, U_{ijk} = r_{ijk} g \quad (6)$$

### 3. Aggregating

The author ID<sub>ij</sub> generates each aggregate signature  $(U_{ij0}, U_{ij1}, \dots, U_{ijm}, U_{ijm+1})$  by:

$$\omega_{ij} = \sum \omega_{ijk} + U_{ij0}, U_{ij1}, \dots, U_{ijm}, U_{ijm+1} \quad (7)$$

When an editor creates a new work using the author's original content, the editor also generates a signature. In this manner, as the number of editors increase, so does the number of signatures. One disadvantage is that the amount of signature data is large. Therefore, signatures are aggregated to reduce this.

### 4. Linking content

Attach the edit management signature if editing is enabled, and attach the hash value if editing is not permitted. Furthermore, attach  $U_{ijk}$ ,  $bID = aID$ . The aggregate signature is attached to the T.B content.

**Verification:** The verification of the content file is performed by the playback device before viewing the content. It is also executed in secondary use. The individual that verifies the signature of the content is called the verifier. If viewing content, the verifier is the viewer. The verifier performs the following processes: Generates the hash value of each partial content for each edit. Prepares the key of the authors and verifies the following equations: The content is accepted as valid if the results of the examinations are accurate.

The verifier examines whether the content is in succession. Then generates the hash value using the parameters and is examined as follows:

$$e(g, \omega_{ij}) = \prod e(U_{ijk}, h_{ijk}) e(g_{pub}, Q_{ij}) \quad (8)$$

If the above validation is successful, the verifier can view or use the content as legitimate content. Content files that fail to be verified cannot be used because they have been illegally edited. This is consistent with the will of the author.

## 5 PERFORMANCE EVALUATION

A computer simulation was performed using the Oracle VM Virtual Box. The Oracle Virtual Box develops a virtual machine in the main computer. Ubuntu software was installed on a virtual computer and the program was executed. Table 1 presents the details of the simulation environment.

The simulation compiled a program generated in C++ using gcc (version 5.4.0) and ran it on the ubuntu terminal. Furthermore, a library called mcl (Herumi, 2020) was introduced to perform finite field operations, elliptic curve calculations, and pairing calculations required for signatures.

### 5.1 Simulation Environment

A computer simulation was performed using the Oracle VM Virtual Box. The Oracle Virtual Box develops a virtual machine in the main computer. Ubuntu software was installed on a virtual computer and the program was executed. Table 1 presents the details of the simulation environment.

Table 1: Simulation environment.

Main machine	Simulation Environment
OS	Windows10Pro
CPU	Intel® Core™ i7-6500U CPU 2.50GHz
RAM	8GB
Virtual machine	Simulation Environment
OS	Ubuntu6.01
CPU	Allocate 1 core from the main CPU
RAM	4GB

The simulation compiled a program generated in C++ using gcc (version 5.4.0) and ran it on the ubuntu terminal. Furthermore, a library called mcl was introduced to perform finite field operations, elliptic curve calculations, and pairing calculations required for signatures.

#### 5.1.1 mcl

mcl is a pairing-based cryptographic library provided by Shigeo Mitunari. Using this library, optimal pairing can be performed on two types of curves: the BN curve and the BLS12-381 curve. mcl is a library that can operate not only on Windows and Linux but also on IOS and Android environments. We used this library on the ubuntu terminal to create and verify signatures.

A feature of the mcl library is that it is faster than the conventional elliptical curve and the pairing arithmetic libraries. For example, 3.9 ms were required by the mcl library to execute on an elliptic curve on arm64. University of Tsukuba Elliptic Curve and Pairing Library (TEPLA), which is another library, required 17.9 ms to calculate another elliptic curve in the same environment (Laboratory of Cryptography and Information Security, 2020). Therefore, mcl has faster processing capabilities than elliptic curve calculation libraries such as RELIC PRIME=254 and MIRACL ake12bnx. We ran the program after compiling it and measured the time required for signature and verification.

### 5.1.2 TEPLA

TEPLA is a library that can implement a cryptographic protocol using pairing in the C language. This library is provided by the Laboratory of Cryptography and Information Security University of Tsukuba and can be used in operating systems such as Windows, Mac OS, and Linux. TEPLA supports calculations on finite fields, elliptic curves, and pairing. TEPLA version 2.0 was released on December 20, 2015.

## 5.2 Outline of Simulation

In this simulation, the 3D model of Miku Miku Dance was used as the content file. This content file is created by the creator (the first user) and signed. Next, there is an editor who secondarily uses this content file. The creator sets the edit control as a parameter. Make sure these settings are followed during secondary use. In addition, suppose some content has been changed even though "Edit prohibited" is set. If it is contrary to the intention of the author, check whether it can be detected as malicious content.

To evaluate the performance of the proposed algorithm, the time required for signature processing was compared to the processing time required by TEPLA. We also measured the processing time required to verify the signature. Table 3 presents the details of the simulation environment using the TEPLA library for comparison. As shown, there is a single 3D model of Miku Miku dance that must be signed, and the signature is generated after the parameters are created by the author.

Figure 4 shows the operation of the user and the operation of the program when signing the content file. User operations are shown in yellow, and program operations are shown in green.

Table 2: Environment TEPLA Ver.2.0.

Main machine	Simulation Environment
OS	Windows10 Home 64bit
CPU	Intel® Core™ i7-9700U CPU
RAM	16GB

Virtual machine	Simulation Environment
OS	Ubuntu6.01
CPU	Allocate 1 core from the main CPU
RAM	4GB

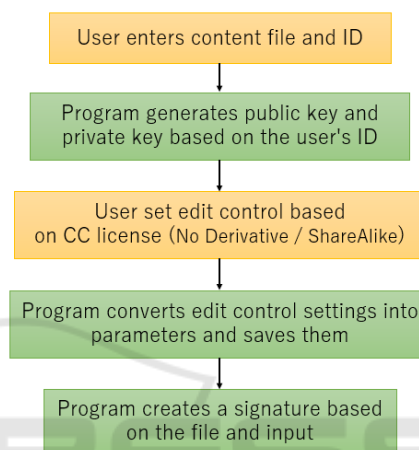


Figure 4: Signature generation flowchart.

## 5.3 Simulation Result

### 5.3.1 Generating Signature

The time required to generate a signature for one 3D model was measured using the mcl library. Table 3 presents the time required to generate the signature using TEPLA. The display time is the average of 5 trials (Fujisaki et al, 2018).

Table 3: Time to generate signature.

Library	Time[s]
mcl	0.1163
TEPLA	0.2153

Based on the measurement, it can be observed that the signature generation time is shorter when mcl is used compared to TEPLA. Therefore, the proposed algorithm was confirmed to be adequately compatible for practical use owing to its high speed.

### 5.3.2 Verification

The time required for signature verification was measured using the mcl library.

Table 4: Time to verify signature.

Library	Time[s]
mcl	0.2972

The processing speed required for signature verification is also practical.

## 6 CONCLUSIONS

A program was implemented to protect the “Miku Miku Dance” content files using ID-based signatures via the mcl library. This algorithm permits secondary use but does not prevent the distribution of the original contents on UGC. Editing can also be completed by secondary users based on inheritance. This ensures that copyright protection is compatible with UGC growth. In addition, the processing speed increases using the mcl library. Furthermore, the developed program can be run on a Linux terminal that, however, hinders observations.

In the future, we plan to incorporate a more user-friendly interface for ease of use.

## REFERENCES

- Inamura, M., Iwamura, K., Watanabe, R., Nishikawa, M., and Tanaka, T., 2011. *a new tree-structure-specified multisignature scheme for a document circulation system*, SERYPT
- Tanaka, D., Iwamura, K., Inamura, M., and Kaneda, K., 2017. *Deployment of Contents Protection Scheme using Digital Signature*, 14<sup>th</sup> International Conference on e-Business
- Fujisaki, M., Iwamura K., Inamura, M., and Kaneda, K., 2018. *Improvement and Implementation of digital content protection scheme using identity based signature*, Fourth International Conference on Mobile and Secure Services.
- Lin, C. Y., Wu, T. C., and Zhang, F., 2003. *A structured multi signature scheme from the gap Diffie Hellman group*, Cryptology ePrint Archive
- Boneh, D., Gentry, C., Lynn, B., and Shacham, H., 2003. *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Advances in Cryptology–EUROCRYPT2003, LNCS, volume 2656, pages 416-432. Springer-Verlag.
- Boneh, D., Gentry, C., Lynn, B., and Shacham, H., 2001. *Short Signatures from the Weil Pairing*, Advances in Cryptology–ASIACRYPT 2001, LNCS, volume 2248, pages 514-532. Springer-Verlag.
- “Youtube” <https://www.youtube.com/> (reading:2020-09-11)
- “Niko Niko Douga” [www.nicovideo.jp/](http://www.nicovideo.jp/) (reading:2020-09-11)

- “Creative Commons License” <https://creativecommons.org/> (reading:2020-09-16)
- Higuchi, Y., “VPVP” <https://sites.google.com/view/vpvp/> (reading:2020-09-11)
- Laboratory of Cryptography and Information Security, University of Tsukuba, “TEPLA” <http://www.cipher.risk.tsukuba.ac.jp/tepla/index.html> (reading:2020-09-11)
- “The GTK+ Project” <https://www.gtk.org/> (reading:2020-09-16)
- Herumi, “mcl” <https://github.com/herumi/mcl> (reading:2020-09-11)