Plant Biotechnology 31, 241–248 (2014)
DOI: 10.5511/plantbiotechnology.14.0609b

Original Paper

A watermarking system for labeling genomic DNA

Naoki Yamamoto, Hiroyuki Kajiura, Shinya Takeno, Nobuaki Suzuki, Yoshihisa Nakazawa*

Hitz Research Alliance Laboratory, Graduate School of Engineering, Osaka University, Suita, Osaka 565-0871, Japan
*E-mail: hitzbiolab@hitzbio.com Tel & Fax: +81-6-6879-4165

Received March 17, 2014; accepted June 9, 2014 (Edited by K. Aoki)

Abstract We established a DNA watermarking system for discriminating transgenic plants. The system contains an encryption algorithm based on a binary system, genetic transformation and a detection algorithm for encrypted DNA watermark sequences using a DNA dot plot. The encryption algorithm converted character strings into nucleic acid sequences through binary digits, and the sequence was designed to be resistant to transition mutations to decipher codes completely. Moreover, the encrypted sequences were capable of taking specific nucleotide sequences in using the algorithmic redundancy of the corresponding DNA. Genetic transformation enables labeling plant genomes with DNA watermarks. The detection algorithm allows finding traces of sequence changes in DNA watermarks, complementing the error protection function of the encryption algorithm. To validate the effectiveness of our DNA watermarking system, we introduced a DNA watermark to the tobacco genome and detected the DNA watermark in PCR products amplified from the genome. This indicates that DNA watermark technology is useful for introducing artificial genetic markers in plant organisms, in particular when several transgenic host plants and transgenes are used. The source codes of the Perl scripts are available in this report.

Key words: Binary digits, decoding, encryption, transgenic plant, watermark.

Much attention has been paid to DNA watermarks for authentication and identification of organisms (Heider and Barnekow 2007). DNA watermark technology employs DNA sequences with encrypted information to label organisms (Arita and Ohashi 2004; Clelland et al. 1999; Gehani et al. 2000; Heider and Barnekow 2008; Heider et al. 2009; Leier et al. 2000; Wong et al. 2003). Recently, DNA watermark technology was tested in an applied study in bacteria and yeast (Heider et al. 2008). It was also used to label an artificial genome in a bacterial study (Gibson et al. 2010). Such applications could be useful for plant breeding. Since farms and plant breeding fields are often established as open-ended systems, plants with valuable genetic traits often face the risk of being pirated without a legally valid mark. Tai et al. (2013) proposed that the use of DNA watermarks could facilitate the authentication and annotation of important plant variety rights. However, to our knowledge, utilization of DNA watermark technology in plants has not previously been reported.

DNA watermark technologies are generally comprised of three processes: encryption, labeling and detection (Halvorsen and Wong 2012; Jupiter et al. 2010). Encryption embeds a particular message or information within a DNA sequence. In labeling, the DNA sequence containing the encrypted information is assimilated into a target organism by genetic engineering (Chou et al. 2004; Mercenier and Chassy 1988; Newell 2000). In detection, the hidden information is mined and decrypted from the genomic sequences to obtain the original message. The encryption process is the most crucial step in DNA watermarking because it primarily determines the subsequent processes. Several encryption methods and algorithms have already been reported (Arita and Ohashi 2004; Clelland et al. 1999; Gehani et al. 2000; Gupta and Singh 2013; Halvorsen and Wong 2012; Heider and Barnekow 2007; Heider and Barnekow 2008; Heider et al. 2009; Jacob and Murugan 2013; Leier et al. 2000; Smith et al. 2003; Tai et al. 2013; Wong et al. 2003). Of these, text-based methods are representative and their characteristics have been well discussed, such as economic advantages and error protection (Smith et al. 2003).

In order to use DNA watermarks for plant breeding, security and error protection are considered to be the most crucial elements of their technologies. Integration of confidential information in targeted breeding lines is one of the main purposes for utilizing watermarks. A hard-to-crack code should be utilized to protect hidden information and the complexity of DNA coding enhances

Abbreviations: GC, guanine and cytosine; PCR, polymerase chain reaction.
This article can be found at http://www.jspcmb.jp/
Published online August 23, 2014

the difficulty in deciphering secret codes by a third party. An error-protecting function is another important element to maintain the encrypted massage for long periods properly in plant cells.

A DNA dot plot is a graphical technique often utilized in the field of bioinformatics. It allows the comparison of two DNA sequences and can identify regions of close similarity between them (Gibbs and McIntyre 1970; Huang and Zhang 2004). One DNA sequence is put on the *x*-axis, and another is put on the *y*-axis. In cases that both sequences contain a region of identical sequence, dots are drawn like a continuous line at the corresponding position. When the region has mismatched bases, insertions, deletions, etc., the region is drawn as a line with missing or discontinuous parts on the plot. Therefore, such a plot can easily be used to detect local similarity of two DNA sequences and has been used to find changes in objective DNA sequences (Maizel and Lenk 1981). DNA dot plots are suitable for finding traces of DNA watermarks when the DNA watermark sequences have changed due to mutational events such as transposon insertions or UV-induced DNA deletions. It is also useful to reinforce error protection in encryption methods.

To establish a DNA watermarking system for breeding and protecting plant varieties, we developed encryption and detection algorithms for DNA watermarks. The encryption algorithm designed DNA watermark sequences with high flexibility and low redundancy at the DNA sequence level, and the detection algorithm mined DNA watermark sequences in a given DNA sequence, representing them on a dot plot. We provided a software tool written in Perl for automation of the encryption and detection. We utilized our watermarking system in tobacco and confirmed that it stably maintained the hidden information in the genome. It was sufficient for discriminating transgenic plant lines. This is the first report of a DNA watermarking system experimentally tested in plant cells.

Materials and methods

Encryption algorithm

A text-based encryption algorithm was developed. The scheme, which converts a character string to a DNA sequence, is illustrated in Figure 1. The algorithm comprises the following four procedures. First, each letter in the character string was assigned to a binary digit. Here we denote the number of binary digits '*n*'. '*n*' was determined according to the number of letters used in the character string, in which the number of letters was greater than 2^{n-1} and less than or equal to 2^n. Second, a code table dedicated to the character string was created. Each letter was randomly allocated to a generated binary string with *n* digits. Third, the character string was converted into a binary digit string according to the code table. Finally, the binary

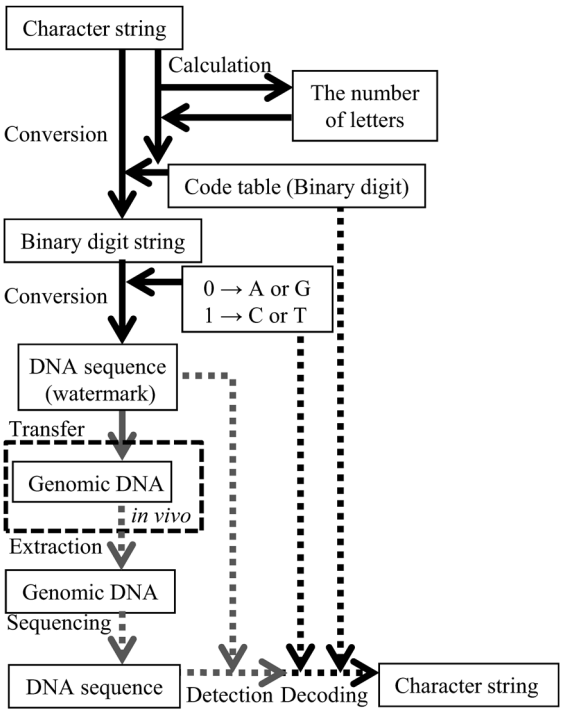

Figure 1. Schematic representation of the DNA watermarking system. The encryption, labeling, detection and decoding processes are shown by solid black line, solid gray line, broken gray line and broken black line, respectively. The box surrounded by a broken black line indicates in vivo processes.

string was converted into a DNA sequence as follows: each "0" was converted into a purine base (A or G), and each "1" was converted into a pyrimidine base (C or T). A or G and C or T were selected at random.

Since selection of A or G and C or T is free, the nucleic acid composition (GC and purine content) of the DNA watermark sequence can be adjusted. Specific DNA sequences can also be embedded in the DNA watermark sequence using the algorithmic redundancy described above. The initial parameters of GC ratio, purine ratio and embedding of specific DNA sequences were set, and the encryption process was designed to generate and adjust DNA watermark sequences repeatedly until the sequence satisfied the initial setting parameters.

Detection algorithm

The detection algorithm allows finding a DNA watermark sequence from a genomic sequence. The algorithm was created using a modification of a dot matrix method (Gibbs and McIntyre 1970; Huang and Zhang 2004). Identical or similar short sequence units were searched by comparing the original DNA watermark sequence with the DNA sequence obtained from genomic DNA (input sequence). The length of the short sequence unit and the acceptance of mismatch number were defined as *k* and *e*, respectively, and the sequence units extracted from DNA watermark sequence were searched against the input sequence. The identical or similar sequence units with a mismatch below *e* were plotted on a graph with the DNA watermark sequence on the *x*-axis and the input

sequence on the *y*-axis. Matched and mismatched sequences were represented on the graph as an asterisk (*) and a cross (+), respectively.

Design of DNA watermark and integration into the tobacco genome

As an example of a character string, "HITZJ001A1" was converted to a DNA watermark sequence. The GC and purine base content were set to 0.4±0.1 and 0.5±0.1, respectively. Bases AAA as an initiation signal and TTT as a termination signal were added to the beginning and end of the DNA watermark sequence, respectively.

The DNA watermark designed, the cauliflower mosaic virus *35S* promoter, and cDNA encoding a prenyltransferase gene (GenBank accession number BAB16687) (Suzuki et al. 2012) were integrated into the backbone of binary vector pCAMBIA2301 (http://www.cambia.org/daisy/cambia/1105.html) to construct a binary plasmid for Agrobacterium-based transformation. First, a DNA fragment containing *35S* promoter sequence was amplified by PCR using the primer set 5′-TCT AGA GTC GAC CTG CAG GC-3′ and 5′-CCT TAG TCA GTC GGT CAC CCC ATG GAA GGA TCT CAA GAA GCC TGT CCT C-3′ using the vector as a template. The PCR product was digested with restriction enzymes *Pst*I and *Bst*EII (New England Biolabs Japan Inc., Tokyo, Japan) and ligated into the same restriction enzyme site of vector pCAMBIA2301. Second, the prenyltransferase gene was amplified by PCR using primer set 5′-GCA ATC AAT TGT CCA TGG CGG AAC TGA AGA AAG AAT TT-3′ and 5′-GTC AAA CTT GGT CAC CAA AAA GAG CAT AAGAAA GCT TCG CCG AAT TCT GTT TCT ACT TGA GCC TCC TGT G-3′. The latter primer was designed to contain the DNA watermark sequence adjacent to the 3′ terminus. The PCR product was digested with *Bst*EII and *Nco*I (New England Biolabs) and ligated downstream of the *35S* promoter. Finally, the resultant vector was electroporated into *Agrobacterium* strain LBA4404 (Hoekema et al. 1984) using a Gene Pulser Xcell electroporation system (Bio-Rad Laboratories Inc., Hercules, CA, USA).

Nicotiana tabacum L. var Xanthi nc. seeds were sterilized and soaked on Murashige & Skoog medium including vitamins (Duchefa Biochemie B. V., Haarlem, The Netherlands) supplemented with 3% sucrose and 0.8% agar under conditions of 16 h light, 8 h dark at 28°C. The plants were grown under sterile conditions, and expanded leaves were excised and used for genetic transformation according to the method described by Horsch et al. (1985). Shoots that regenerated on medium supplemented with 50 μg l^{-1} kanamycin were excised from calli and transferred onto rooting medium (Horsch et al. 1985). Rooting shoots were planted on Metro-Mix 350 soil (Sun Gro Horticulture Canada Ltd., Vancouver, Canada) and acclimatized in an indoor phytotron chamber. The plantlets were grown under conditions of 16 h light, 8 h dark at 28°C for over 5 weeks; young leaves were used for genomic DNA analysis.

Detection of DNA watermarks in tobacco

Tobacco leaves were ground into powder with a mortar and pestle in liquid nitrogen. Genomic DNA was extracted from the powder using a NucleoSpin Plant II kit (Takara Bio Inc., Otsu, Japan). Primers, 5′-CCC AGA ATC TGT TGC TAG AGT GAA-3′ and 5′-CAA GAC CGG CAA CAG GAT TC-3′, corresponding to 182–158 bp upstream of the start site and 61–80 bp downstream of the end site of the DNA watermark sequence, respectively, were used for PCR with genomic DNA. The resultant PCR products were analyzed by agarose gel electrophoresis. HyperLadder I (Bioline Ltd., London, UK) was used as a DNA molecular weight marker. Wild-type tobacco plants were used as a negative control. After electrophoresis, the PCR products were purified from agarose gels using a FastGene Gel/PCR Extraction kit (Nippon Genetics Co., Ltd., Tokyo, Japan) and directly sequenced using the primer 5′-CCC AGA ATC TGT TGC TAG AGT GAA-3′.

The DNA watermark of the sequence data was analyzed using a homemade Perl script that implemented the detection algorithm. The initial parameters were unit length $k=8$, number of mismatch sequences $e=1$. The output file for the dot matrix plot was viewed by a text editor, Sakura Editor (http://sakura-editor.sourceforge.net).

Results

DNA watermarking system

The DNA watermarking system is comprised of four components: encryption, labeling, detection and decoding. A character string is encrypted into a DNA watermark sequence via a binary digit string. A DNA fragment with the DNA watermark is transferred to a host genome to label the host organism. The watermarked DNA is extracted with the host genomic DNA and sequenced. The sequencing results are searched to find the DNA watermark. The detected DNA watermark is restored to its original character string in the decoding process.

To design an encryption algorithm suitable for plant breeding, a binary system was employed as an intermediate to encrypt character strings into DNA watermark sequences. The scheme of the algorithm is shown in Figure 1. In the initial step, all letters in a given character string were assigned to randomly produced binary strings of n digits with one-to-one correspondence. To minimize the length of the watermark sequences, the number of digits (n) of the binary digit sequence was determined each time according to the number of letters in the character string. A code table dedicated to the character string was created each time, and the character string was converted into a binary string using the table. The binary string created was sequentially converted into a DNA sequence, in which "0" and "1" were respectively converted into random purine and pyrimidine bases. The randomized

correspondence between the binary digits and nucleotide bases resulted in flexibility of the generated DNA watermark sequences in terms of nucleotide (GC and purine) composition and sequence patterns.

For automating the encryption process, the algorithm was implemented in a computer tool written in Perl (Supplemental Figure S1). At the top of the source code, a given character string and some initial parameters, like GC content, purine content and particular embedded sequences, were set. According to the parameters, the input character string was converted into a DNA watermark sequence, and the generated code table was also output as a text file with the watermark sequence. The encryption software tool generated a 33-mer DNA watermark sequence in 1 s on a laptop computer intended for daily use (2.4 GHz dual-core processor and 4 GB memory).

Detection of DNA watermarks

Direct sequencing of PCR products is one of the most convenient methods for examining sequences at a specific region of genomic DNA. To facilitate detection of DNA watermark sequences, the PCR products were sequenced directly, and a detection algorithm using a DNA dot plot was developed to implement as a Perl script. This algorithm can represent matched/

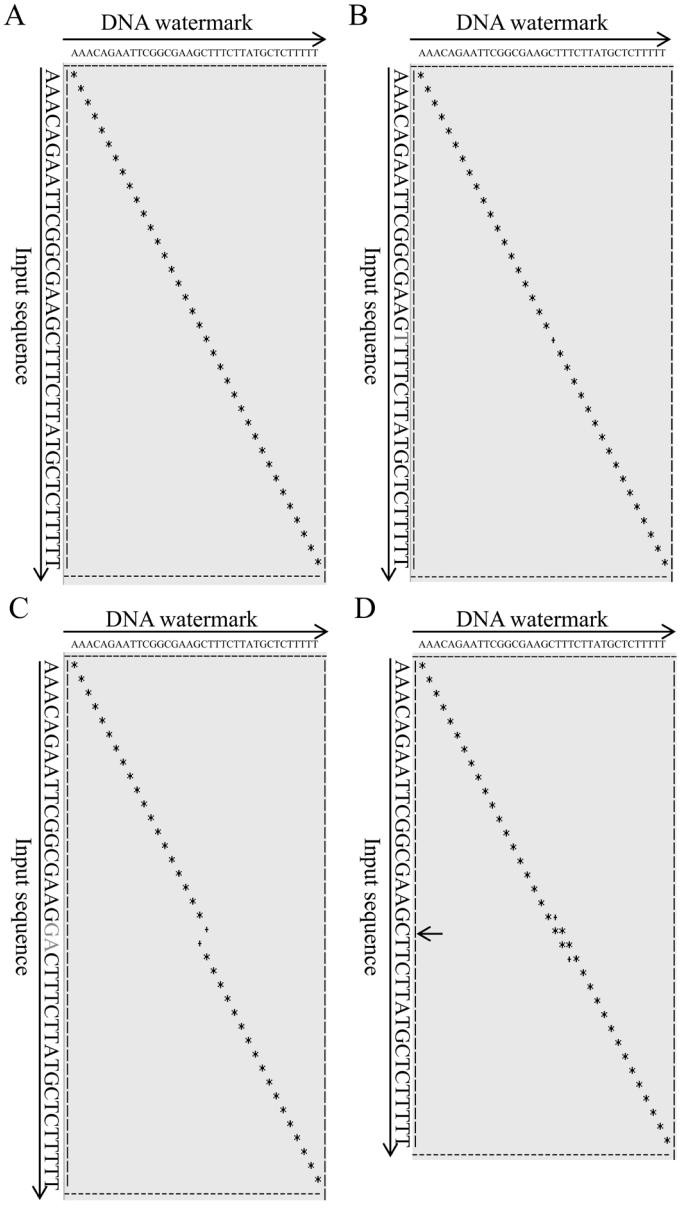

Figure 2. Detection of a DNA watermark on a DNA dot plot. DNA watermark sequences and input sequences are shown along the horizontal and vertical axes, respectively. Asterisk (*) indicates the two sequences are identical in a given region. A cross (+) represents mutated sequences. (A) The case of complete identity between the DNA watermark and the input sequence. (B) The case of a transition mutation occurring in the middle of the input sequence. (C) The case of a two-base insertion occurring in the middle of the input sequence. (D) The case of a one-base deletion in the middle of the input sequence. Mutation and insertion sequences are represented in gray. Deletion site is indicated by an arrow.

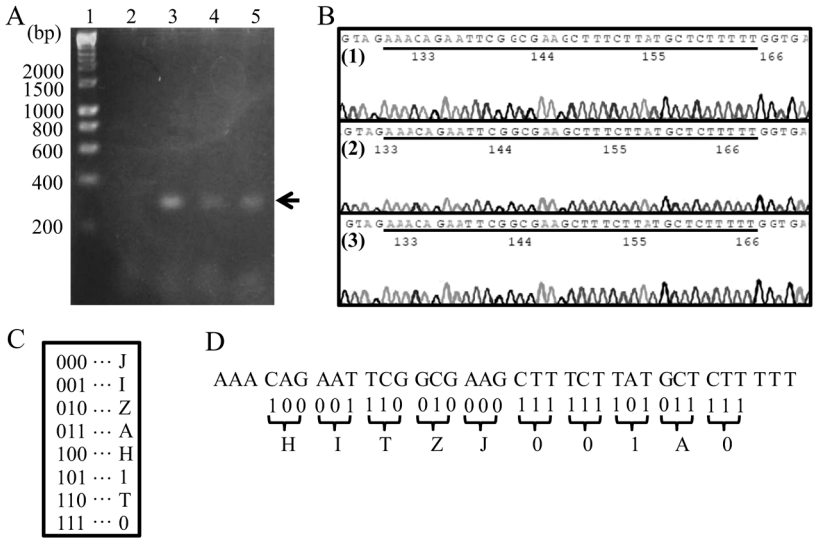

Figure 3. Detection and decryption of the DNA watermark sequence from tobacco. (A) Agarose gel electrophoresis of the PCR products amplified from the T-DNA region including the DNA watermark in transgenic tobacco. PCR products were separated in an 0.8% gel. Lane 1, DNA molecular weight marker; lane 2, wild-type tobacco; lanes 3–5, transgenic tobacco lines 1–3. (B) Sequencing chromatograms of the PCR products in the transgenic tobacco. DNA watermark sequences ware underlined. The numbers on the left side (1)–(3) indicate transgenic tobacco lines 1–3, respectively. (C) Code table of the DNA watermark. (D) Decryption of the DNA watermark detected in transgenic tobacco.

mismatched regions of two DNA sequences as a graphical view of a two-dimensional matrix as a straight diagonal line (Figure 2A). Even when a mutation, insertion and deletion were introduced to the input sequence, traces of the DNA watermark were observed as diagonal lines (Figure 2B, C, D). The detection tool generated an output file in 1 s on a laptop computer. The output file was viewed by a text editor such as Sakura Editor.

A model study in tobacco

In order to test whether DNA watermarks designed by this system function as intended, a DNA watermark sequence was generated and integrated into tobacco. The character string "HITZ001A0" was used as an example and converted into a DNA watermark sequence with the initiation and termination codes AAA and TTT, respectively. The DNA watermark was cloned into a binary vector with a prenyltransferase gene driven by the *35S* promoter and introduced into the tobacco genome by Agrobacterium-mediated transformation. After generation of transgenic tobacco plants, three independent lines were randomly selected, and the genomic DNA regions including the DNA watermark sequence were amplified by PCR. A single band was detected in all of the transgenic lines on an agarose gel (Figure 3A). The size of the detected PCR products was approx. 300 bp, almost identical to the estimated length of the PCR products, which was 297 bp. The PCR products were purified and sequenced, and the data obtained were identical to the original DNA watermark sequence (Figure 3B). The sequence was manually

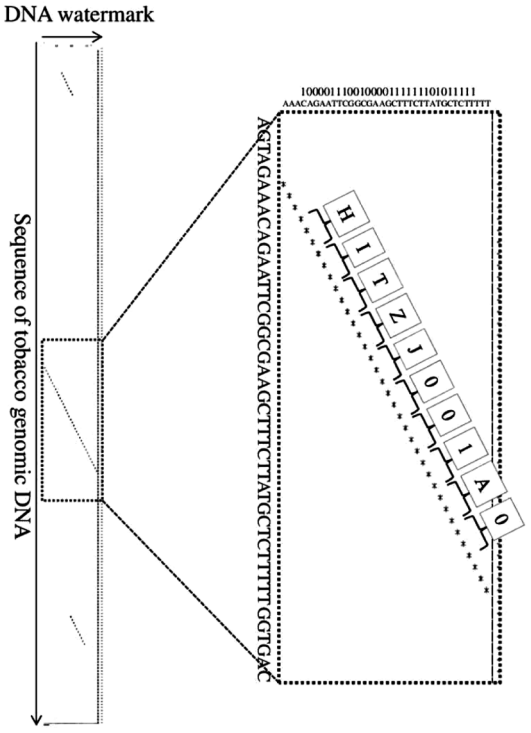

Figure 4. Detection of the DNA watermark in transgenic tobacco on a DNA dot plot. DNA watermark sequence and input sequence are shown along the horizontal and vertical axis, respectively. Asterisk (*) indicates identity between the two sequences in the region. The dot plot area of the DNA watermark sequence is magnified on the right. The numbers shown along the top of the magnified image are binary digit strings corresponding to the DNA watermark. The detected sequence was decoded into the character string according to the code table (shown in Figure 3C).

decoded to the character string based on a code table (Figure 3C, D, Figure 4). The encrypted information in the form of a DNA watermark sequence was successfully restored from its transgenic tobacco genomic sequence.

Discussion

Discrimination of genetically valuable organisms is essential to manage breeding. However, development of line-specific genetic (or phenotypic) markers to discriminate each bred line is often arduous. Recently, DNA watermarking has been recognized as a reliable technology to label breeding lines. Organisms harboring DNA watermarks in their genome can be easily discriminated at the molecular level (Heider and Barnekow 2007). DNA watermarking technology is also required for protection of useful bred lines from theft (Jupiter et al. 2010). Previously, several watermarking technologies were developed for microbial genomes (Arita and Ohashi 2004; Heider and Barnekow 2008). In contrast, there are few reports of DNA watermarking technology for plants. Therefore, we developed a simple DNA watermarking system to label plant genome for utilizing it as an artificial genetic marker. This system was comprised of four components, encryption, labeling, detection and decoding, and worked sufficiently in the model plant tobacco.

Recently, application of DNA watermarks for authentication of plant variety rights was proposed (Tai et al. 2013). That study presented a scheme to hide a message in an mRNA sequence using the genetic code. The scheme utilizes the substitution of synonymous codons and enables integration of a DNA watermark into open reading frames without functional changes to the encoded protein. Moreover, it is easy to crack due to its employment of the genetic code, which has simple and well-known rules in the biosciences. Nevertheless, use of the genetic code has two problems. One is incomplete decoding when nonsynonymous substitution at the DNA level occurred. Nonsynonymous substitutions are an important source for breeding. Another is that its security level is low.

Our DNA watermarking system has high flexibility due to its binary-based encryption system with a code table created each time. To enhance the stability of DNA watermarks in the target genome, we utilized flexibility and concealment during design of the sequence to adapt fundamental DNA sequence characteristics to the genome. In fact, the GC content of plant genomic DNA is quite divergent among species (Smarda and Bures 2012). In addition, this flexibility allows embedding functional sequences in DNA watermarks to expand the potential usefulness of DNA watermarking. Such a characteristic can be applied to detection of a DNA watermark by simple detection using restriction enzymes or transgene regulation. For example, creating a recognition site of a restriction enzyme by using this flexibility could make possible to detect the existence of the DNA watermark sequence in genome with ease. The DNA fragment encoding a watermarking sequence can be amplified by PCR and checked by digestion using the enzyme. With regard to transgene regulation, a DNA watermark with *cis*-regulatory sequences could be designed in the promoter region of transgene constructs. To date, various short sequences involve in gene regulation in plant tissues were collected (Higo et al. 1999).

Error protection in DNA watermarks is required for utilization in plants. Plants are often exposed to environments triggering DNA mutation during growth, breeding and propagation, such as UV irradiation, radiation, chemical mutagens and somaclonal variation (Larkin and Scowcroft 1981; Evans 1989). Moreover, some of these mutations can become fixed in the breeding process or during evolution if the mutations are not deleterious (Barton and Keightley 2002). Walker et al. (2006) described bud sports often being observed and used for improving productivity or quality of agricultural production. As one countermeasure against mutation, Smith et al. (2003) described a code comprised of sixty-four 6-base codons based on the order of purine and pyrimidine bases. Similarly, our encryption algorithm is based on the purine and pyrimidine bases. The advantage of such a method is that successful decoding of DNA watermarks is independent of the influence of transition mutations. Even though the potential for transversion mutations, insertions and deletions remains, the DNA dot plot detection algorithm addresses this issue. In this study, since stability of the DNA watermark was checked only in one generation, the stability and robustness of DNA watermarks designed by this encryption algorithm remains to be tested over several generations.

Another aspect of this encryption algorithm is that it first converts character strings to sequences of binary numbers based on a dedicated code table. Since the number of digits used in creating a code table can be arbitrarily determined, a wide variety of letters can be allocated to binary strings and utilized in a character string. Therefore, a variety of types of information could be encrypted. In our model for DNA watermarking, information on owner, transformed host cultivar and transgene constructs was written using only the Latin alphabet and numerals (Figure 5A); however, other alphabets, such as Greek, Arabic, or Japanese, can also be utilized by allocating binary numbers (Figure 5B). When several genes or plant lineages are used for systematic transformation, transformed plants need to be screened and managed based on their traits. For systematic use in breeding, a watermark sequence describing a variety of information involving their traits is advisable to simplify working progress and to avoid mistakes due to artifacts.

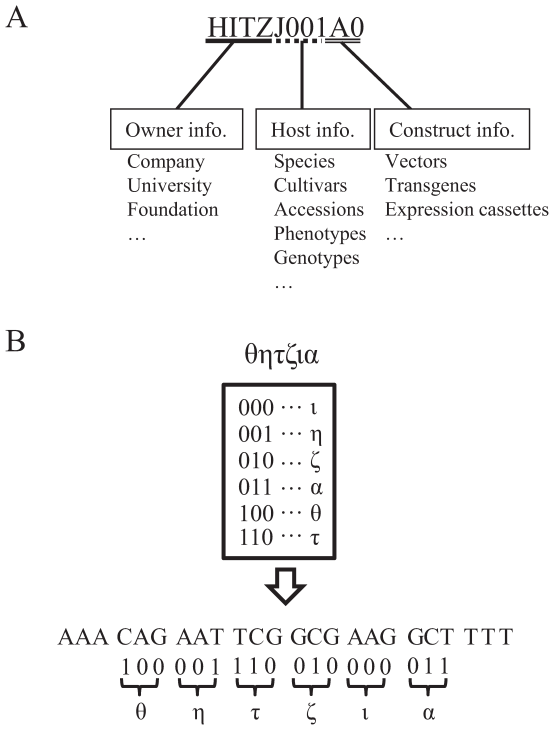

Figure 5. (A) A model of the DNA watermark designed in this study. Owner, host and genetic construct information are included in character strings that are converted into DNA watermarks. (B) Another example of a character strings, "*θητζια*" was converted to a DNA watermarking sequence. The letters of Greek alphabet were allocated to binary strings.

One potential use of our DNA watermarking system is for labeling artificial plant genomes. Gibson et al. (2010) presented the use of watermarks for a chemically synthesized bacterial genome to distinguish it from other genomes. In plants, advanced technologies have been developed to transfer large DNA fragments (Shibata and Liu 2000). These technologies can transfer dozens of tandemly connected genes into plant genomes to generate modified chromosomal DNA (Dafny-Yelin and Tzfira 2007). In transformation of multiple genes, varieties of gene sets are often thought to be used. For discrimination of generated transgenic plant lines, assignment of a series of DNA watermarks would be effective for their management when many genetic constructs are used. Thus, this application would accelerate objective selection of progeny after crossing different transgenic lines.

The encryption and detection tools were written in Perl, and work readily on commodity-type computers. Therefore, this DNA watermarking approach could be adapted for routine use, opening the possibility of utilizing DNA watermarks in plants for bioindustrial applications.

Acknowledgements

This work was partially supported by the New Energy and Industrial Technology Development Organization (NEDO).

References

Arita M, Ohashi Y (2004) Secret signatures inside genomic DNA. *Biotechnol Prog* 20: 1605–1607

Barton NH, Keightley PD (2002) Understanding quantitative genetic variation. *Nat Rev Genet* 3: 11–21

Chou TH, Biswas S, Lu S (2004) Gene delivery using physical methods: An overview. *Methods Mol Biol* 245: 147–166

Clelland CT, Risca V, Bancroft C (1999) Hiding messages in DNA microdots. *Nature* 399: 533–534

Dafny-Yelin M, Tzfira T (2007) Delivery of multiple transgenes to plant cells. *Plant Physiol* 145: 1118–1128

Evans DA (1989) Somaclonal variation: Genetic basis and breeding applications. *Trends Genet* 5: 46–50

Gehani A, LaBean T, Reif J (2000) DNA-based cryptography. *DIMACS Ser Discrete Math Theoret Comput Sci* 54: 233–249

Gibbs AJ, McIntyre GA (1970) The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. *Eur J Biochem* 16: 1–11

Gibson DG, Glass JI, Lartigue C, Noskov VN, Chuang RY, Algire MA, Benders GA, Montague MG, Ma L, Moodie MM, et al. (2010) Creation of a bacterial cell controlled by a chemically synthesized genome. *Science* 329: 52–56

Gupta K, Singh S (2013) DNA based cryptographic techniques: A review. *International Journal of Advanced Research in Computer Science and Software Engineering* 3: 607–610

Halvorsen K, Wong WP (2012) Binary DNA nanostructures for data encryption. *PLoS ONE* 7: e44212

Heider D, Barnekow A (2007) DNA-based watermarks using the DNA-Crypt algorithm. *BMC Bioinformatics* 8: 176

Heider D, Barnekow A (2008) DNA watermarks: A proof of concept. *BMC Mol Biol* 9: 40

Heider D, Kessler D, Barnekow A (2008) Watermarking sexually reproducing diploid organisms. *Bioinformatics* 24: 1961–1962

Heider D, Pyka M, Barnekow A (2009) DNA watermarks in non-coding regulatory sequences. *BMC Res Notes* 2: 125

Higo K, Ugawa Y, Iwamoto M, Korenaga T (1999) Plant *cis*-acting regulatory DNA elements (PLACE) database: 1999. *Nucleic Acids Res* 27: 297–300

Hoekema A, Roelvink PW, Hooykaas PJ, Schilperoort RA (1984) Delivery of T-DNA from the *Agrobacterium tumefaciens* chromosome into plant cells. *EMBO J* 3: 2485–2490

Horsch RB, Fry JE, Hoffmann NL, Eichholtz D, Rogers SG, Fraley RT (1985) A simple and general method for transferring genes into plants. *Science* 227: 1229–1231

Huang Y, Zhang L (2004) Rapid and sensitive dot-matrix methods for genome analysis. *Bioinformatics* 20: 460–466

Jacob G, Murugan A (2013) DNA based cryptography: An overview and analysis. *Int J Emerg Sci* 3: 36–42

Jupiter DC, Ficht TA, Samuel J, Qin QM, de Figueiredo P (2010) DNA watermarking of infectious agents: Progress and prospects. *PLoS Pathog* 6: e1000950

Larkin PJ, Scowcroft WR (1981) Somaclonal variation: A novel source of variability from cell cultures for plant improvement. *Theor Appl Genet* 60: 197–214

Leier A, Richter C, Banzhaf W, Rauhe H (2000) Cryptography with DNA binary strands. *Biosystems* 57: 13–22

Maizel JV Jr, Lenk RP (1981) Enhanced graphic matrix analysis of nucleic acid and protein sequences. *Proc Natl Acad Sci USA* 78: 7665–7669

Mercenier A, Chassy BM (1988) Strategies for the development of bacterial transformation systems. *Biochimie* 70: 503–517

Newell CA (2000) Plant transformation technology. Developments and applications. *Mol Biotechnol* 16: 53–65

Shibata D, Liu YG (2000) Agrobacterium-mediated plant transformation with large DNA fragments. *Trends Plant Sci* 5: 354–357

Smarda P, Bures P (2012) The variation of base composition in plant genomes. *Plant Genome Diversity* 1: 209–235

Smith GC, Fiddes CC, Hawkins JP, Cox JP (2003) Some possible codes for encrypting data in DNA. *Biotechnol Lett* 25: 1125–1130

Suzuki N, Uefuji H, Nishikawa T, Mukai Y, Yamashita A, Hattori M, Ogasawara N, Bamba T, Fukusaki E, Kobayashi A, et al. (2012) Construction and analysis of EST libraries of the *trans*-polyisoprene producing plant, *Eucommia ulmoides Oliver. Planta* 236: 1405–1417

Tai WL, Wang CCN, Sheu PCY, Tsai JJP (2013) Data hiding in DNA for authentication of plant variety rights. *JESTC* 11: 38–42

Walker AR, Lee E, Robinson SP (2006) Two new grape cultivars, bud sports of Cabernet Sauvignon bearing pale-coloured berries, are the result of deletion of two regulatory genes of the berry colour locus. *Plant Mol Biol* 62: 623–635

Wong PC, Wong KK, Foote H (2003) Organic data memory using the DNA approach. *Commun ACM* 46: 95–98

```
# !/usr/bin/perl ;

# Initial parameter setting

# Correspondance between nucleic acids and the binary digit.
$an = 0 ; # A is 0
$gn = 0 ; # G is 0
$cn = 1 ; # C is 0
$tn = 1 ; # T is 0

$gccont = 0.4 ; # GC content of watermarks generated
$gccontmargin = 0.1 ; # The margin of GC content
$princont = 0.5 ; # purine content of watermarks generated
$princontmargin = 0.1 ; # The margin of purine content

# If users would like to take paticular sequences in watermarks,
# the sequences can be input as follows.
$fseq{1} = $fseqori{1} = gaattc ; # The first one (paticular sequences)
$fseq{2} = $fseqori{2} = aagctt ; # The second one (paticular sequences)

# The option for initiation and termination signal ofthe cipher
$iniseq = aaa ; # Initiation signal
$endseq = ttt ; # Termination signal

$messageori = YAMAMOTON ; # Input character string

foreach $key (keys %fseq) {
    while (1) {
    if ($fseq{$key} =~ /^(.)(.+)$/) {
        $temp1 = $1 ;
        $temp2 = $2 ;
        if ($temp1 eq a || $temp1 eq A || $temp1 eq g || $temp1 eq G) {
            $intseqnum{$key} = $intseqnum{$key}."0" ;
        } elsif ($temp1 eq c || $temp1 eq C || $temp1 eq t || $temp1 eq T) {
            $intseqnum{$key} = $intseqnum{$key}."1" ;
        }
        $fseq{$key} = $temp2 ;
    } elsif ($fseq{$key} =~ /^(.)$/) {
        $temp1 = $1 ;
        if ($temp1 eq a || $temp1 eq A || $temp1 eq g || $temp1 eq G) {
            $intseqnum{$key} = $intseqnum{$key}."0" ;
        } elsif ($temp1 eq c || $temp1 eq C || $temp1 eq t || $temp1 eq T) {
            $intseqnum{$key} = $intseqnum{$key}."1" ;
        }
        last ;
    }
    }
}
$messagelength = length($messageori) ;
$message = $messageori ;
$count = 0 ;
for ($x = 1; $x <=$messagelength ; $x++) {
    $flag0 = 0 ;
    while (1) {
    if ($message =~ /^(.)(.+)$/) {
        $temp1 = $1 ;
        $temp2 = $2 ;
        for ($a = 1; $a <=$count ; $a++) {
            if ($temp1 eq $word{$a}) {
                $flag0 = 1 ;
                $message = $temp2 ;
                last ;
            }
        }
        if ($flag0 ne 1) {
            $count++ ;
            $word{$count} = $temp1 ;
        } else {
            last ;
        }
        $message = $temp2 ;
    } elsif ($message =~ /^(.)$/) {
        $temp1 = $1 ;
        for ($a = 1; $a <=$count ; $a++) {
            if ($temp1 eq $word{$a}) {
                last ;
            }
        }
            if ($flag0 ne 1) {
                $count++ ;
                $word{$count} = $1 ;
            }
        }
        last ;
    }
}

for ($b = 0; $b <=$count ; $b++) {
    if ($messagelength >= (2**$b) && $messagelength < (2**($b+1))) {
        $keta = ($b) ;
    }
```

```
}

while(1) {
    %numtotal = () ;
    for ($i = 1; $i <=$count ; $i++) {
        $flag = 0 ;
        $numtotal = "" ;
        for ($c = 1; $c <=$keta ; $c++) {
            $num = int(rand(2)) ;
            $numtotal = $numtotal.$num
        }
        for ($j = 1; $j <=$i ; $j++) {
            if ($numtotal eq $numtotal{$j}) {
                $flag = 1 ;
                last ;
            }
        }
        if ($flag eq 1) {
            redo ;
        }
        $numtotal{$i} = $numtotal ;
    }
    open (OUT2, ">./codeseq.txt") ; # A code table was output to a file (codeseq.txt)
        for ($i = 1; $i <=$count ; $i++) {
            print OUT2 "$numtotal{$i}¥t$word{$i}¥n" ;
        }
    close (OUT2) ;
    $message = $messageori ;
    $seqnum = "" ;
    while (1) {
        if ($message =~ /^(.)(.+)$/) {
            $temp1 = $1 ;
            $temp2 = $2 ;
            for ($xx = 1; $xx <=$count ; $xx++) {
                if ($temp1 eq $word{$xx}) {
                    $seqnum = $seqnum.$numtotal{$xx} ;
                }
            }
            $message = $2 ;
        } elsif ($message =~ /^(.)$/) {
            $temp1 = $1 ;
            for ($xx = 1; $xx <=$count ; $xx++) {
                if ($temp1 eq $word{$xx}) {
                    $seqnum = $seqnum.$numtotal{$xx} ;
                }
            }
            last ;
        }
    }
    $seqnumout = $seqnum ;
    $flagcheck1 = 0 ;
    foreach $key (keys %fseq) {
        if ($seqnum =~ /$intseqnum{$key}/) {
            $seqnum =~ s/$intseqnum{$key}// ;
        } else {
            $flagcheck1 = 1 ;
        }
    }
    if ($flagcheck1 eq 0) {
    } else {
        $seqnum = "" ;
        next ;
    }
    $seqnum = $seqnumout ;
    $seqnumlength = length ($seqnum) ;
    $numcount = 0 ;
    $numcountp = 0 ;
    $flagprin = 0 ;
    while (1) {
        if ($seqnum =~ /^(.)(.+)$/) {
            $temp1 = $1 ;
            $seqnum = $2 ;
            if ($temp1 eq 0) {
                $numcountp++ ;
            }
        } elsif ($seqnum =~ /^(.)$/) {
            $temp1 = $1 ;
            if ($temp1 eq 0) {
                $numcountp++ ;
            }
            $flagprin = 1 ;
        }
        if ($flagprin eq 1) {
            last ;
        }
    }
    if ((($numcountp / $seqnumlength) < ($princont - $princontmargin)) || (($numcountp / $seqnumlength) > ($princont + $princontmargin))) {
        $seqnum = "" ;
        next ;
    }
    $seqnum = $seqnumout ;
```

```
$flagcheck2 = 0 ;

foreach $key (keys %fseqori) {
    if ($seqnum =~ /$intseqnum{$key}/) {
        $seqnum =~ s/$intseqnum{$key}/$fseqori{$key}/ ;
    } else {
        $flagcheck2 = 1 ;
    }
}
print "$seqnum¥n" ;
$loopcount = 0 ;
$seq = "" ;
while (1) {
    $loopcount++ ;
    if ($loopcount >1000) {
        $flagredu = 1 ;
        last ;
    }
    if ($seqnum =~ /^(.)(.+)$/) {
        $temp1 = $1 ;
        $seqnum = $2 ;
        if ($temp1 eq 0) {
            $num = int(rand(2)) ;
            if ($num eq 0) {
                $seq = $seq."A" ;
            } elsif ($num eq 1) {
                $seq = $seq."G" ;
            }
        } elsif ($temp1 eq 1) {
            $num = int(rand(2)) ;
            if ($num eq 0) {
                $seq = $seq."C" ;
            } elsif ($num eq 1) {
                $seq = $seq."T" ;
            }
        } else {
            $seq = $seq.$temp1 ;
        }
        next ;
    } elsif ($seqnum =~ /^(.)$/) {
        $temp1 = $1 ;
        $num = int(rand(2)) ;
        if ($temp1 eq 0) {
            $num = int(rand(2)) ;
            if ($num eq 0) {
                $seq = $seq."A" ;
            } elsif ($num eq 1) {
                $seq = $seq."G" ;
            }
        } elsif ($temp1 eq 1) {
            $num = int(rand(2)) ;
            if ($num eq 0) {
                $seq = $seq."C" ;
            } elsif ($num eq 1) {
                $seq = $seq."T" ;
            }
        } else {
            $seq = $seq.$temp1 ;
        }
    }
        if ((length($seq) ne $seqnumlength)) {
            next ;
        }
        $seqout = $seq ;
        $seqcountp = 0 ;
        $flaggc = 0;
        while (1) {
            if ($seq =~ /^(.)(.+)$/) {
                $temp1 = $1 ;
                $seq = $2 ;
                if ($temp1 eq G || $temp1 eq C || $temp1 eq g || $temp1 eq c) {
                    $seqcountp++ ;
                }
            } elsif ($seq =~ /^(.)$/) {
                $temp1 = $1 ;
                if ($temp1 eq G || $temp1 eq C || $temp1 eq g || $temp1 eq c) {
                    $seqcountp++ ;
                }
                $flaggc = 1 ;
            }
            if ($flaggc eq 1) {
                last ;
            }
        }
        if ((($seqcountp / $seqnumlength) < ($gccont - $gccontmargin)) || (($seqcountp / $seqnumlength) > ($gccont + $gccontmargin))) {
            $seq = "" ;
            next ;
        } else {;
            last ;
        }
```

```
    }
    if ($flagredu eq 1) {
        next ;
    }
    last ;
}
open (OUT1, ">./watermark.txt") ; # Output file (watermark.txt) of watermark sequences
print "$seqout¥t($numcountp / $seqnumlength)¥n" ;
print OUT1 "$seqout¥n" ;
print OUT1 "" ;
print OUT2 "" ;
close (OUT1) ;
```

```
# !/usr/bin/perl ;

# Initial parameter setting
$startseq = "AAA" ; # The signal of the initiation of the cipher
$endseq = "TTT" ; # The signal of the termination of the cipher
$watermark = "CAGAATTCGGCGAAGCTTTCTTATGCTCTT" ; # The sequence for searching
$framelen = 8 ; # Unit length k
$error = 1 ; # Acceptance of the number of errors in the unit

open (IN, "sequence.txt") or die; # The sequence file in text format (sequence.txt)
while (<IN>) {
    chomp ;
    $seqall = $seqall.$_ ;
}
close (IN) ;
$seqlength = length($seqall) ;

$watermarkall = $startseq.$watermark.$endseq ;
$watermarklength = length($watermarkall) ;
for ($x = 0; $x <= ($watermarklength - $framelen) ; $x++) {
    $partwatermark = substr ($watermarkall, $x, $framelen) ;
    $partwatermarkori = $partwatermark ;
    while (1) {
        $partwatermark = $partwatermarkori ;
         $countmiss = 0 ;
        for ($z = 0; $z <= ($seqlength - $framelen) ; $z++) {
          $judge = 1 ;
            $partseq = substr ($seqall, $z, $framelen) ;
            for ($i = 0; $i <= ($framelen - 1) ; $i++) {
                $watermarkchara = substr ($partwatermark, $i, 1) ;
                $partseqchara = substr ($partseq, $i, 1) ;
                if ($watermarkchara eq n) {
                } elsif ($watermarkchara eq $partseqchara) {
                } else {
                    $countmiss++ ;
                }
                if ($countmiss > $error) {
                    $judge = 0 ;
                    $countmiss = 0 ;
                    last ;
                }
            }
            if ($judge == 1) {
                for ($i = 0; $i <= ($framelen - 1) ; $i++) {
                    $watermarkchara = substr ($partwatermark, $i, 1) ;
                    $partseqchara = substr ($partseq, $i, 1) ;
                    if ($watermarkchara eq n) {
                        if ($flag{$x+$i}{$z+$i} ne 2) {
                            $flag{$x+$i}{$z+$i} = 1 ;
                        }
                    } elsif ($watermarkchara eq $partseqchara) {
                        if ($flag{$x+$i}{$z+$i} ne 1) {
                            $flag{$x+$i}{$z+$i} = 2 ;
                        } else {
                            $flag{$x+$i}{$z+$i} = 1 ;
                        }
                    } else {
                        if ($flag{$x+$i}{$z+$i} ne 2) {
                            $flag{$x+$i}{$z+$i} = 1 ;
                        } else {
                            $flag{$x+$i}{$z+$i} = 1 ;
                        }
                    }
                }
            }
        }
        $partwatermark = $partwatermarkori ;
        last ;
        if ($flagpoint{$framelen - 2} eq 1 && $flagpoint{$framelen - 1} eq 1) {
            last ;
        }
    }
}

# To standard output of the result
print "¥n" ;
$premaxx = length($watermarkall) ;
$maxx = int(($premaxx * 10**(-1))) / 10**(-1) ;
print "0" ;
for ($aa = 1; $aa <= $premaxx ; $aa++) {
    $aaa = $aa % 10 ;
    if ($aaa == 9) {
        $write++ ;
        print "$write" ;
    } elsif ($aaa == 0) {
        print "0" ;
    } elsif ($aa >= ($maxx+1)) {
        print "¥n" ;
        last ;
    } else {
        print " " ;
```

```
    }
}
for ($aaa = 1; $aaa <= ($premaxx+2) ; $aaa++) {
    print "_" ;
}
print "¥n" ;

for ($bb = 0; $bb <= ($seqlength-1) ; $bb++) {
    print "|" ;
    for ($aaa = 0; $aaa <= ($premaxx-1) ; $aaa++) {
        if ($flag{$aaa}{$bb} == 0) {
            print " " ;
        } elsif ($flag{$aaa}{$bb} == 1) {
            print "+" ;
        } elsif ($flag{$aaa}{$bb} == 2) {
            print "*" ;
        }
    }
    print "|¥n" ;
}
for ($aaa = 1; $aaa <= ($premaxx+1) ; $aaa++) {
    print "-" ;
}
    print "|¥n" ;
```