Dynamic Matrices from Hardware Fingerprint and Its Application in Software Copyright Protection

Ning Huang

Center of Modern Educational Technology, Gannan Normal University
Ganzhou, 341000, China
hngzjx@qq.com

Abstract

This paper presents a scheme of software copyright protection based on dynamic matrices from hardware fingerprint and user's information. It is mainly as follows. (1)Introduces the basic idea of software copyright protection; (2)Introduces the alphabetic operation of field $\boldsymbol{F}_{37}$; (3)Uses hardware fingerprint and user's information to create dynamic matrices to design an affine mapping of $\boldsymbol{F}_{37}$ to generate a registration code; (4) Uses the invert mapping to verify a registration code. (5)Gives concrete examples. Data and keys are used in a way to prevent sharing registration code with any other computer. The text submitted to the vendor by the user is not the source plaintext for the registration code. Real plaintext is hidden in the software, unknown to the outsiders. It is deceptive to possible adversaries. Attacks for a legal registration code from the submitted text will not success. Experimental results and analysis illustrate the algorithm viable and secure.

Keywords: *Copyright, Fingerprint, Matrices, Protection, Software, Registration*

1. Introduction

Software copyright piracy can be a worldwide sticky question. This encourages vendors to develop various methods against it. In general, the concept of software includes all kinds of electronic documents.While in particular, this concept includes computer programs. Der-Chyuan et al.[1, 2] proposed a two-phase watermarking scheme to protected images and another scheme for digital image copyright protection. Wei-Huang et al.[3] proposed a copyright protection method for digital image with $1/T$ rate forword error correction(FEC). Ester et al.[4] propose a new algorithm based on Harr discrete wavelet transform for the grayscale watermarking. H.-H. Tsai et al.[5] presented a robust lossless watermarking technique, based on a-trimmed mean algorithm and support vector machine (SVM), for image authentication. Ibrahim et al.[6] advocated protecting software copyright through hiding watermarks in various data structures. Tzung-Her et al.[7, 8] analyzed the security of a robust copyright protection scheme based on visual cryptography proposed by Lou et al., also a lightweight copyright protection protocol involving secret-key cryptosystems and a tamper-resistant device. Aimin et al.[9] proposed a scheme of security dog to protect software copyright. Mao et al.[10] discussed principle and technique of software copyright protection. Ning et al.[11, 12, 13, 14] proposed ideas of dynamic constitution of matrix from cyclic group first used in the area of copyright protection, a multi-scale triangular mapping to protect the copyright and permission control of software copyright by dynamic constitution of Vandermonde matrix in extended Hill's cryptosystem [15].

This paper concerns the protection of programs downloaded from the Internet. Such kind of software can be controlled into several levels according to the designer. The permission control is usually performed by a registration system which produces registration codes of different level of permission[13, 14]. A generic method using invertible matrix from hardware fingerprint along with the users information is also used.

The rest of the paper is organized as follows. Section 2 briefly introduces permission control based on registration system. Section 3 proposes the algorithm of software copyright protection. Section 4 gives experimental results and analysis. Section 5 concludes the paper.

2. Permission Control Based on Registration System

Fig.1 Registration Pattern illustrates a frequently used registration process, which will be discussed as follows.

2.1 With User's Changeable Information

Having paid the necessary fee, the user sends personal information to the vendor via network or another tunnel. The vendor encrypts the user's information (plaintext) into registration code (ciphertext, sometimes called activation code) and sends it back to the user. After the registration code is keyed in, verification program is invoked by the application system to check the legitimacy of the registration code. This program decrypts the ciphertext and compares it with the user's information which has been preliminarily keyed in by the user. The successful comparison permits the user's

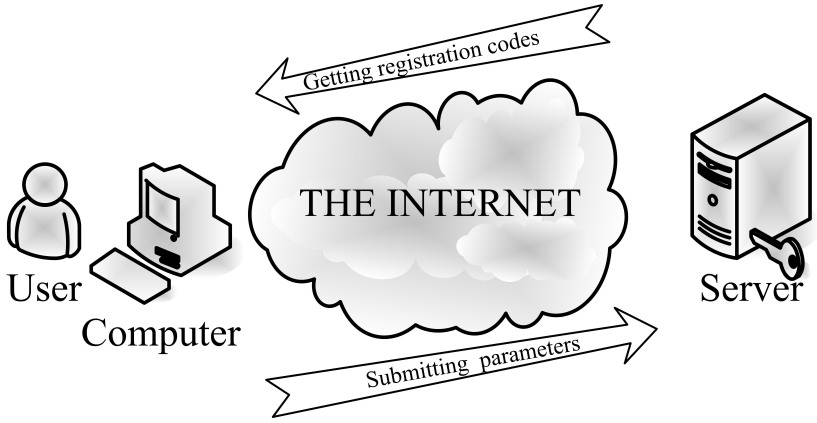

Fig.1 Registration Pattern.

registration and the user gets the permission to use the software. The main problem in this scheme is that any valid user's registration code along with the user's information can be used in any other computer, because the scheme is independent to any other information.

2.2 With User's Unchangeable Information

The user's unchangeable information involved in the encryption mainly comes from the user's hardware such as software dog [9], hard drive serial number, hard disk partition volume number, machine access code. Being easy to bring about conflicts with other peripherals and other problems, software dog is hard to be popularly used. Both hard disk partition volume number and machine access code can be modified in the system, so they are not the favorite hardware identifications. Hard disk serial number is unchangeable and it can be used as a unique identification[10]-[13].We recommend that hard disk serial be used as fingerprint of user's hardware. Having paid the necessary fee, the user sends fingerprint relevant information to the vendor via network or another tunnel. The vendor encrypts the user's information (plaintext) into registration code (ciphertext) and sends it back to the user. After the registration code is keyed in, verification program is invoked by the application system to check the legitimacy of the registration code. This program decrypts the ciphertext and compares it with the user's information which is relevant to the fingerprint. The successful comparison permits the user's registration and the user gets the permission to use the software. The advantage of the method is that it can prevent plagiarism of registration from any other legal user.

2.3 Withstanding Algebraic Attacks

To improve the copyright protection, we propose a scheme of encrypting the plaintext which is hidden within the software as system parameter by the dynamic constitution of invertible matrices from fingerprint to form a registration code, so that the relationship between fingerprint and registration code is not of plaintext and ciphertext and it can withstand algebraic attacks based on registration code calculation from the known fingerprint. Existing algebraic attacks for a legal registration string from the fingerprint will not success.

3. The Algorithm of Software Copyright Protection

3.1 Designing Ideas

1) We are going to generate matrices dynamically from fingerprint for encryption function.Deferent computers generate deferent matrices. Thus, every encrypt key is only useful to a single computer and we get a high level of security;
2) In the user's program, the verifying module can only generate matrices for decryption, while the matrices for encryption to calculate the registration code can only be generated by the vendor's program;
3) The fingerprint is a unique string highly dependent on a hard disk serial number. The user submits fingerprint to the vendor to generate the encryption matrices to calculate the registration code from a preliminary string .The registration string is send back to the user.
4) Registration code is the result from a preliminary string rather than the fingerprint. Any attempt to calculate the registration code from the fingerprint will not success.
5) When the user runs the application, decryption matrices are generated dynamically from fingerprint, checking the legitimacy of the registration code. The registration can be done and stored on the user's computer if and only if the verification is approved.
6) Each time when the user starts the application, it will first verify the registration code to determine the legitimacy and the user's permission to use the application software, for example, professional version, standard version, trial version, demo version or refuse to run.

3.2 Algorithm Conventions

We extend the alphabet of Hill's cryptosystem[15] from 26 letters to 37 [13]. We use the length 37 because it is a prime. So $\boldsymbol{Z}_{37}$ is a Galois field $\boldsymbol{F}_{37}$. Every non-zero element in $\boldsymbol{F}_{37}$ has a unique inverse. The computation of division is closed. Thus, we can avoid the inconvenience of zero factors in a ring. Furthermore, registration system mainly uses capital letters and numbers to create a license string. In a plaintext, we only use the last letter '$' as the ending mark of a string. Similar to Hill's original paper [6], We define $\boldsymbol{F}_{37} = \{a_i | i = 0, ..., 36\}$, in which $a_0 a_1 ... a_{36}$ is a permutation of $0, 1, ..., 9, A, B, ..., Z, \$$ .Without loss of generality, we take $0, 1, ..., 9, A, B, ..., Z, \$$ as $a_0 a_1 ... a_{36}$ and

call i the value of a_i.Without causing confusion, we consider a_i and i the same thing. It follows that we have $0 = 0, ..., 9 = 9, A = 10, ..., Z = 35, \$ = 36$ in our system.
The addition in the field can be carried out by the value addition *modulo* 37, i.e., $a_i + a_j = a_k, where\ k = i + j\ mod\ 37$. Likewise, the multiplication in the field can be carried out by the value multiplication *modulo* 37, i.e., $a_i \cdot a_j = a_k, where\ k = i \cdot j\ mod\ 37$.
Both operations can be expressed in the form of integers as well as letters. For example, we take two arbitrary elements from the $a_0 \cdots a_{36}$, say $K = a_{20}$ and $J = a_{19}$. We calculate $K+J$ by $20+19\ mod\ 37$ and get 2, which determines element $a_2 = 2$. So the calculation can be expressed as $20 + 19 = 2$ or $K + J = 2$, if we have "mod 37" in mind. Likewise, the product of K and J can be expressed as $20 \cdot 19 = 10$ or $K \cdot J = A$.
An integer n is preset. Different non-zero elements $c_1c_2...c_k$ are taken from fingerprint, in which lower case is converted to upper and symbols not in our alphabet are ignored. If $k < n$, then we use the circularly to reach n .If $k \geq n$, we take no more c_is from the fingerprint. In practice, if there are already enough c_is, we can adjust n to an appropriate value to match a number of c_is.
Without loss of generality, a lower triangular invertible matrix C is formed, such that $c_{11}c_{22}...c_{nn}$ are from $c_1c_2...c_k$ circularly and $c_{ij} = 0$ for $i < j$ and $c_{ij} = a_i$ for $i > j$, along with a lower triangular invertible $L = L_n = (l_{ij})$ and an upper triangular invertible H, such that $l_{ij} = a_i$ for $i = j$, $l_{ij} = a_i + a_j$ for $i > j$ and $l_{ij} = 0$, for $i < j$ and $H = L^T$(the transpose of L).
For matrices C, L and H, inverses are easy to be obtained recursively (refer to the **Theorem** in **Appendix**). Let $A = LCH$. Then C is "hidden" in A by L and H.
This idea is similar to (but different from) that of multivariate public key system[16, 17, 18, 19].Matrix C defines a linear mapping.If we substituted this linear mapping with a multi-variate polynomial mapping, the scheme might become a multi-variate one.However, multi-variate system is not the topic of our discussion in the paper and we would rather set it aside now.
A long enough string P, such as
$P = PROFESSIONALVERSION$,
is used to control the permission.P will be put into an $n \times x$ matrix $P_{n\times x}$vertically.If there are some blank cells in the last column, "$" will be used as ending mark.
Suppose the user's personal information $U = u_1u_2...u_r$, a matrix $U_{n\times x} = (u_{ij})_{n\times x}$ will be created, such that $u_{11}...u_{n1}u_{12}...u_{n2}...u_{1n}...u_{nn}$ are from $u_1u_2...u_r$ circularly, with the same size as $P_{n\times x}$

3.3 Generation of Registration Code (Encryption Algorithm)

System preliminary parameters: Presets a permission string P and positive integer n .
Input: Fingerprint like $c_1c_2...c_k$, user's personal data like $U = u_1u_2...u_r$.
Output: Registration string like
$V = xxxxxx - xxxxxx - \cdots - xxxxxx$.

Step 1 Fingerprint $c_1c_2...c_k$ is taken by the application and sent to the vendor via network or another tunnel along with $U = u_1u_2...u_r$;

Step 2 The vendor puts the fingerprint into a program called registration code generator;

Step 3 Different non-zero elements $c_1c_2...c_k$ are taken by the generator. If $k < n$, $c_1c_2...c_k$ are taken circularly until the number reaches n .If $k \geq n$, the process stops at n .In the end a sequence $c_1c_2...c_n$ is obtained;

Step 4 Constitutes matrices C, L, H dynamically as mentioned in 3.2;

Step 5 Computes the invert matrices C^{-1}, L^{-1}, H^{-1} respectively;

Step 6 Computes $A^{-1} = H^{-1}C^{-1}L^{-1}$ to obtain the encryption matrix;

Step 7 Puts the preliminary string P into a matrix $P_{n\times x}$ .If there are some blank elements in the last column of the matrix when the string meet its end, fills in the blanks with "$"s to stand for the end;

Step 8 Constitutes matrix $U_{n\times x}$ as mentioned in 3.2 to match the size of $P_{n\times x}$;

Step 9 Computes $V_{n\times x} = A^{-1}P_{n\times x} + U_{n\times x}$, takes the elements from by columns, each column forms a segment. Uses hyphen "−" to connect the segments to form a string like
$V = xxxxxx - xxxxxx - \cdots - xxxxxx$
and sends back to the user as a registration code.

3.4 Registration Verification (Decryption)

Preliminary string is defined by the system just the same as that in 3.3. On the user's side, the application checks whether there is a registration code in the system. If there is a registration code in the system, the program checks the legitimacy by decrypting the ciphertext into plaintext and comparing the result with the preliminary string. If the comparison is successful, the application can be run properly and the registration code can be stored on the computer if this is the first successful running of the application. If there is no registration code or the registration code is false, the user cannot get the normal right to run the application and is offered a form to register.
Input: Registration code like
$V = xxxxxx - xxxxxx - \cdots - xxxxxx$
(from keyboard or storage of the system) and user's personal information U ;
Output: P' ;

Step 1 Takes fingerprint, gets elements $c_1c_2...c_n$, constitutes matrix dynamically;

Step 2 Ignores connector "−", puts V into a matrix $V_{n\times x}$;

Step 3 puts U into a matrix of $U_{n\times x}$ to match the size of $V_{n\times x}$;

Step 4 Computes $P_{n\times x} = A(V_{n\times x} - U_{n\times x})$;

Step 5 Takes elements from $P_{n\times x}$ to form a string P' in which "$" stands for end;

Step 6 Compares P' with P. If the comparison is a success, the user gets the corresponding right to use the software and the application program is invoked and next step is not carried out;

Step 7 Prompts the user to register the software. The user can only choose to register again or exit the program.

4. Experimental Results and Analysis

Preliminary setting in the system: Embeds version control string

$$P = PROFESSIONALVERSION$$

and dimension number $n = 6$.

4.1 Registration Code Generation

Step 1 Fingerprint $S07GJ10Y$ is taken by the application and sent to the vendor via network or another tunnel along with the user's personal information, for example, $U = Alice$;

Step 2 The vendor puts the fingerprint $S07GJ10Y$ and $U = Alice$ into a program called registration code generator;

Step 3 Takes 6 different non-zero elements $c_1 = S, c_2 = 7, c_3 = G, c_4 = J, c_5 = 1, c_6 = Y$;

Step 4 Constitutes matrices dynamically C, L, H as mentioned in 3.2;

$$C = \begin{pmatrix} S & 0 & 0 & 0 & 0 & 0 \\ 1 & 7 & 0 & 0 & 0 & 0 \\ 1 & 2 & G & 0 & 0 & 0 \\ 1 & 2 & 3 & J & 0 & 0 \\ 1 & 2 & 3 & 4 & 1 & 0 \\ 1 & 2 & 3 & 4 & 5 & Y \end{pmatrix},$$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 3 & 0 & 0 & 0 \\ 4 & 5 & 6 & 4 & 0 & 0 \\ 5 & 6 & 7 & 8 & 5 & 0 \\ 6 & 7 & 8 & 9 & A & 6 \end{pmatrix},$$

$H = L^T$;

Step 5 Computes the invert matrices C^{-1}, L^{-1}, H^{-1} respectively,

$$C^{-1} = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ A & G & 0 & 0 & 0 & 0 \\ H & Z & 7 & 0 & 0 & 0 \\ Z & M & W & 2 & 0 & 0 \\ 7 & Y & \$ & T & 1 & 0 \\ Y & X & B & E & E & C \end{pmatrix},$$

$$L^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \$ & J & 0 & 0 & 0 & 0 \\ P & O & P & 0 & 0 & 0 \\ 9 & 5 & I & S & 0 & 0 \\ A & H & P & 7 & F & 0 \\ N & L & 9 & 8 & C & V \end{pmatrix},$$

$H^{-1} = (L^{-1})^T$;

Step 6 To obtain the encryption matrix, Computes

$$A^{-1} = H^{-1}C^{-1}L^{-1} = \begin{pmatrix} 6 & F & \$ & 9 & 4 & 9 \\ Z & G & W & R & H & 5 \\ B & D & 9 & 6 & 9 & I \\ 6 & 5 & E & 1 & E & G \\ 3 & G & 3 & 3 & X & O \\ D & Q & A & \$ & M & P \end{pmatrix};$$

Step 7 Puts the preliminary string P into a matrix $P_{6\times4}$ .There are 5 blank elements in the last column of the matrix when the string meet its end, fills in the blanks with "$" s to stand for the end,

$$P_{6\times4} = \begin{pmatrix} P & S & V & N \\ R & I & E & \$ \\ O & O & R & \$ \\ F & N & S & \$ \\ E & A & I & \$ \\ S & L & O & \$ \end{pmatrix};$$

Step 8 Puts $U = Alice$ into matrix $U_{6\times4}$ as mentioned in 3.2 to match the size of $P_{6\times4}$,

$$U_{6\times4} = \begin{pmatrix} A & L & I & C \\ L & I & C & E \\ I & C & E & A \\ C & E & A & L \\ E & A & L & I \\ A & L & I & C \end{pmatrix};$$

Step 9 Computes

$V_{6\times4} = A^{-1}P_{6\times4} + U_{6\times4} =$

$$\begin{pmatrix} 6 & F & \$ & 9 & 4 & 9 \\ Z & G & W & R & H & 5 \\ B & D & 9 & 6 & 9 & I \\ 6 & 5 & E & 1 & E & G \\ 3 & G & 3 & 3 & X & O \\ D & Q & A & \$ & M & P \end{pmatrix} \begin{pmatrix} P & S & V & N \\ R & I & E & \$ \\ O & O & R & \$ \\ F & N & S & \$ \\ E & A & I & \$ \\ S & L & O & \$ \end{pmatrix} + \begin{pmatrix} A & L & I & C \\ L & I & C & E \\ I & C & E & A \\ C & E & A & L \\ E & A & L & I \\ A & L & I & C \end{pmatrix} = \begin{pmatrix} M & K & 2 & 3 \\ 8 & 6 & Y & K \\ Q & 7 & P & N \\ Y & Y & D & Z \\ X & P & 8 & 8 \\ D & 2 & P & 7 \end{pmatrix},$$

takes the elements from $V_{6\times4}$ by columns, each column forms a segment. Uses hyphen "−" to connect the segments to form a string

$V = M8QYXD - K67YP2 - 2YPD8P - 3KNZ87$

and sends back to the user as a registration code.

Fig.2 Registration Code Generation illustrates the encryption process.

4.2 Verification of Registration Code

Step 1 Takes fingerprint, constitutes matrix $A = LCH$ from $c_1c_2...c_n$ dynamically;

$$A = LCH = \begin{pmatrix} S & J & A & 1 & T & K \\ L & X & 8 & 6 & 4 & 2 \\ H & S & Z & 8 & X & L \\ G & V & 0 & X & Z & V \\ I & F & L & 4 & 6 & A \\ N & H & S & D & E & G \end{pmatrix};$$

Step 2 Puts

$V = M8QYXD - K67YP2 - 2YPD8P - 3KNZ87$

into a matrix $V_{6\times4}$ except connector "−",

$$V_{6\times4} = \begin{pmatrix} M & K & 2 & 3 \\ 8 & 6 & Y & K \\ Q & 7 & P & N \\ Y & Y & D & Z \\ X & P & 8 & 8 \\ D & 2 & P & 7 \end{pmatrix};$$

Step 3 Puts $U = Alice$ into matrix $U_{6\times4}$ as mentioned in 3.2 to match the size of $P_{6\times4}$,

$$U_{6\times4} = \begin{pmatrix} A & L & I & C \\ L & I & C & E \\ I & C & E & A \\ C & E & A & L \\ E & A & L & I \\ A & L & I & C \end{pmatrix};$$

Step 4 Computes

$V_{6\times4} - U_{6\times4}$

$$= \begin{pmatrix} M & K & 2 & 3 \\ 8 & 6 & Y & K \\ Q & 7 & P & N \\ Y & Y & D & Z \\ X & P & 8 & 8 \\ D & 2 & P & 7 \end{pmatrix} - \begin{pmatrix} A & L & I & C \\ L & I & C & E \\ I & C & E & A \\ C & E & A & L \\ E & A & L & I \\ A & L & I & C \end{pmatrix}$$

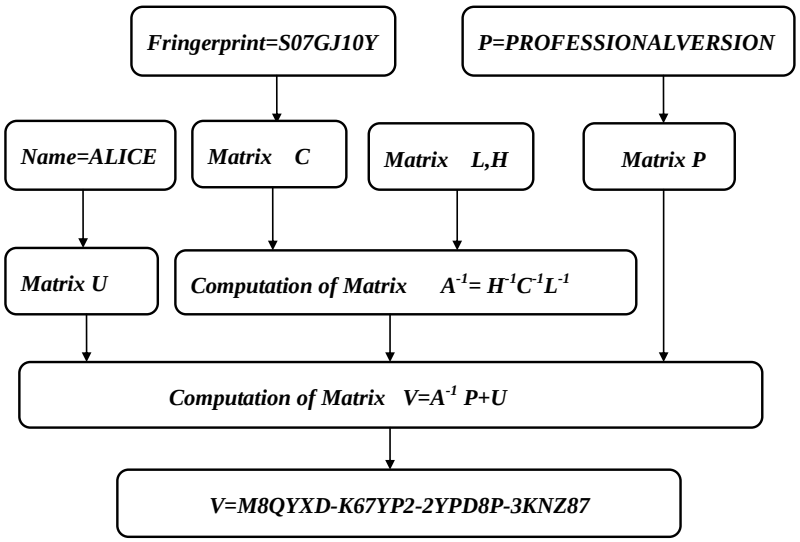

Fig.2 Registration Code Generation.

$$= \begin{pmatrix} C & \$ & L & S \\ O & P & M & 6 \\ 8 & W & B & D \\ M & K & 3 & E \\ J & F & O & R \\ 3 & I & 7 & W \end{pmatrix},$$

$$P_{6\times 4} = A(V_{6\times 4} - U_{6\times 4})$$

$$= \begin{pmatrix} S & J & A & 1 & T & K \\ L & X & 8 & 6 & 4 & 2 \\ H & S & Z & 8 & X & L \\ G & V & 0 & X & Z & V \\ I & F & L & 4 & 6 & A \\ N & H & S & D & E & G \end{pmatrix} \begin{pmatrix} C & \$ & L & S \\ O & P & M & 6 \\ 8 & W & B & D \\ M & K & 3 & E \\ J & F & O & R \\ 3 & I & 7 & W \end{pmatrix}$$

$$= \begin{pmatrix} P & S & V & N \\ R & I & E & \$ \\ O & O & R & \$ \\ F & N & S & \$ \\ E & A & I & \$ \\ S & L & O & \$ \end{pmatrix};$$

Step 5 Takes elements from $P_{6\times 4}$ to form a string $P' = PROFESSIONALVERSION\$\$\$\$\$$ in which "$" stands for end, i.e. $P' = PROFESSIONALVERSION$;

Step 6 Compares P' with P. If the comparison is a success, the user gets the corresponding right to use the software and the application program is invoked and the registration code is stored on the computer and next step is not carried out;

Step 7 Prompts the user to register the software. The user can only choose to register again or exit the program.

Fig.3 Verification of Registration Code illustrates the decryption process.

4.3 Potential Improvements to the Scheme

From the above discussion, we propose that this scheme be enhanced by minor adjustments, for example, we can
1) use *meaningless* string such as $56DETU9E4TFGFBVBGIYU7100W$ instead of using *meaningful* string such as $PROFESSIONALVERSION$;
2)increase the lengthy of permission control string;
3)vary the dimension n according to the fingerprint;
4)include more information such as vendor's information in U.
All these will raise the robustness of the software copyright protection.

5. Conclusions

We use dynamic constitution of matrices from fingerprint and user's information to design the scheme of software copyright protection. Experimental results and analysis illustrate the algorithm viable. Our encryption depends on the matrices dynamically generated from fingerprint. Thus, Encryption varies on different computers. Any attempt to share registration code with any other computer is doomed to failure. The relationship between information submitted by the user and the registration code is not of plaintext and ciphertext. The plaintext is actually hidden within the software, not known to the outsiders, so that existing algebraic attacks are hard to figure out registration code from fingerprint and user's information. Furthermore, the user's program does not create a registration code even in peer mode. If more tricks listed in *4.3* are applied, we can assure the scheme secure.

Appendix

Theorem *In a field* $\boldsymbol{F}$*, a lower triangular matrix*

$$L = (l_{ij}) = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix}$$

with non-zero elements l_{ii} *in the diagonal is invertible and* L^{-1} *can be computed recursively.*

Proof

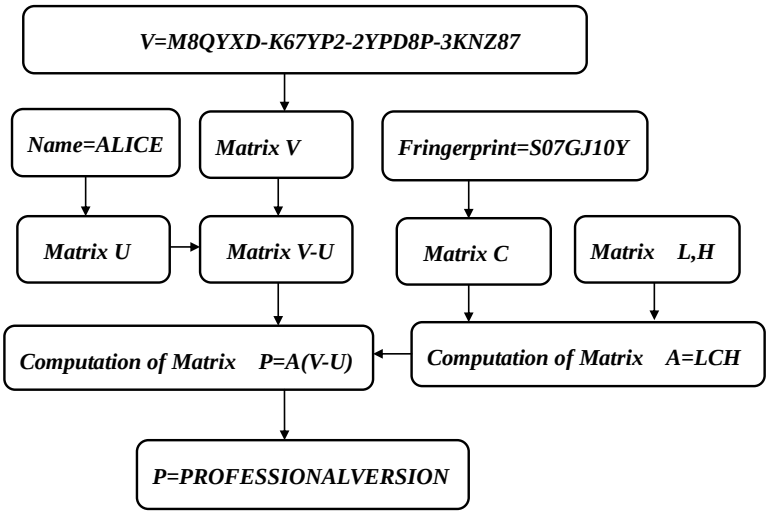

Fig.3 Verification of Registration Code.

1) L is invertible:Obviuosly we have

$$det(L) = \prod_{i=1}^{n} l_{ii}.$$

It follows from $l_{ii} \neq 0$ and $\boldsymbol{F}$ has no zero factor that $det(L) \neq 0$. Thus L is invertible.
2)The recursive formulae to compute L^{-1}:
If $n = 1$, then $L = L_1 = (l_{11})$, we have $L^{-1} = (l_{11}^{-1})$;
If $n = 2$, then $L = L_2 = (l_{ij}) = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix}$. Let $L_2X_2 = I_2$,
where $X_2 = (x_{ij}) = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}$, and $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,
then we have
$l_{11}x_{11} = 1$, $l_{11}x_{21} = 0$, $l_{21}x_{11} + l_{22}x_{21} = 0$,
$l_{21}x_{12} + l_{22}x_{22} = 1$;
thus
$x_{11} = l_{11}^{-1}$, $x_{12} = 0$, $x_{21} = -l_{21}^{-1}l_{11}l_{22}^{-1}$, $x_{22} = l_{22}^{-1}$.
and $\begin{pmatrix} l_{11}^{-1} & 0 \\ -l_{21}^{-1}l_{11}l_{22}^{-1} & l_{22}^{-1} \end{pmatrix}$ is the solution of
$L_2X_2 = I_2$, $L_2^{-1} = \begin{pmatrix} l_{11}^{-1} & 0 \\ -l_{21}^{-1}l_{11}l_{22}^{-1} & l_{22}^{-1} \end{pmatrix}$;
Suppose for $n-1$, L_{n-1}^{-1} is already known.
Let $LX = L_nX_n =$
$\begin{pmatrix} L_{n-1} & 0_{(n-1)\times 1} \\ L_{1\times(n-1)} & l_{nn} \end{pmatrix} \begin{pmatrix} X_{n-1} & X_{(n-1)\times 1} \\ X_{1\times(n-1)} & x_{nn} \end{pmatrix}$
$= \begin{pmatrix} I_{n-1} & 0_{(n-1)\times 1} \\ 0_{1\times(n-1)} & 1 \end{pmatrix} = I_n,$
where L_{n-1}, X_{n-1} and I_{n-1} are blocks of $(n-1)\times(n-1)$ from up-left to down-right of L_n, X_n and I_n respectively;
$L_{1\times(n-1)}$, $X_{1\times(n-1)}$ and
$0_{1\times(n-1)}$ are blocks of $1\times(n-1)$;
$L_{(n-1)\times 1}$, $X_{(n-1)\times 1}$ and
$0_{(n-1)\times 1}$ are blocks of $(n-1)\times 1$;
$0_{1\times(n-1)}$ and $0_{(n-1)\times 1}$ are blocks with all zero elements, I_n is the identity matrix of $n \times n$.
We obtain four equations of matrices:
$L_{n-1}X_{n-1} = I_{n-1}$,
$L_{n-1}X_{(n-1)\times 1} = 0_{(n-1)\times 1}$,
$L_{1\times(n-1)}X_{n-1} + l_{nn}X_{1\times(n-1)} = 0_{1\times(n-1)}$,
$L_{1\times(n-1)}X_{(n-1)\times 1} + l_{nn}x_{nn} = 1$;
By solving these equations, we obtain:
$X_{n-1} = L_{n-1}^{-1}$,
$X_{(n-1)\times 1} = 0_{(n-1)\times 1}$,
$X_{1\times(n-1)} = -L_{1\times(n-1)}L_{n-1}^{-1}l_{nn}^{-1}$,
$x_{nn} = l_{nn}^{-1}$;
so that, we obtain the recursive formulae:
$L_1^{-1} = (l_{11}^{-1})$ and
$L_n^{-1} = \begin{pmatrix} L_{n-1}^{-1} & 0_{(n-1)\times 1} \\ -L_{1\times(n-1)}L_{n-1}^{-1}l_{nn}^{-1} & l_{nn}^{-1} \end{pmatrix}$ for L^{-1}.
According to the principle of mathematical induction, the theorem holds.
Furthermore, from the process of the proof, we can directly deduce the following corollary:
Corollary *If a lower triangular matrix L is invertible, then it's invert matrix L^{-1} is also lower triangular.*

Acknowledgements

1)We would like to express thanks to the hard work of the editors and reviewers of this paper.
2)This work was supported by the fund from Natural Science of Jiangxi Province of China under Grant No.20114BAB201033. We would like to express thanks to the Committee of the fund.

References

[1] M.C. Hu, D.C.Lou and M.C. Changb, "Dual-wrapped digital watermarking scheme for image copyright protection", Computers & Security 26 (2007)319-330.

[2] D.C. Lou, H.K. Tso and J.L. Liu, "A Copyright Protection Scheme for Digital Images Using Visual Cryptography Technique", Computer Standards & Interfaces 29 (2007) 125-131.

[3] W.H. Lin, S.J. Horng, T.W. Kao, R.J. Chen, Y.H. Chen, C.L. Lee and T. Terano,"Image Copyright Protection with Forward Error Correction ", Expert Systems with Applications 36 (2009) 11888-11894.

[4] E. Yen, K.S. Tsai, "HDWT-based grayscale watermark for copyright protection", Expert Systems with Applications 35 (2008) 301-306.

[5] H.H. Tsai, H.C. Tseng and Y.S. Lai, "Robust lossless image watermarking based on α-trimmed mean algorithm and support vector machine", The Journal of Systems and Software 83 (2010) 1015-1028.

[6] I.Kamel and Q.Albluwi, "A robust software watermarking for copyright protection", Computers & Security 28 (2009) 395C409.

[7] T.H. Chen, C.C. Chang, C.S. Wu and D.C. Lou c, "On the security of a copyright protection scheme based on visual cryptography", Computer Standards & Interfaces 31 (2009) 1-5.

[8] T.H. Chen and G.Horng, "A lightweight and anonymous copyright-protection protocol", Computer Standards & Interfaces 29 (2007) 229-237.

[9] A.M. Hou, "Security Dog Helps to Protect the Software Copyright", Journal of Donggan University of Technology, Oct 2005, Vol. 12, No.5, pp.28-32.

[10] M. Tan, Y. Chen and J.Tu, "Study on Software Copyright Protection Technology", Computer Applications and Software, Jan 2007, Vol.24, No.1, pp.64-68.

[11] N. Huang and X.T. Huang, "An Algorithm of Software Copyright Protection Based on Dynamic Constitution of Integer Matrix", International Journal of Communication and Security, July 2010, Vol.1, No.1, pp.14-17.

[12] N. Huang, X.T. Huang and X.W. He, "A New Algorithm of Software Copyright Protection Based on Multi-scale Triangular Mapping", Information Science and Engineering, December 2009, pp.472 - 475

[13] N. Huang, "Permission Control of Software Based on Registration System with Vandermonde Matrix in a Galois Field", In Instrumentation & Measurement, Sensor Network and Automation (IMSNA), 2012 International Symposium on, 2012, Vol. 2, pp.487-490.

[14] N. Huang,X.T. Huang,J.L. Ren,X.W. He and Y. Liu "Further Research on Registration System with Vandermonde Matrix", International Journal of Computer Science Issues, 2013, Vol. 10, Issue 1, pp.614-640.

[15] L.S. Hill, "Cryptography in an Algebraic Alphabet", The American Mathematical Monthly, July 1929, Vol.36, No.6, pp.306-312.

[16] T.Matsumoto and H. Imai,"Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption", Advances in cryptology -EUROCRYPT'88, LNCS, Springer, 1988, vol.330, pp.419-453.

[17] J.T. Ding, "A new variant of the Matsumoto-Imai cryptosystem through perturbation", Public Key Cryptosystems, Springer 2004, vol.2947, pp.305-318.

[18] X. Wang, F. Feng, X.M. Wang and Q. Wang, "A More Secure MFE Multi-variate Public Key Encryption Scheme", International Journal of Computer Science and Applications, 2009, vol.6, No.3, pp.1-9.

[19] N. Courtois, "The security of hidden field equations (HFE)", Advances in Cryptology -ASIACRYPT 2002, CT-RSA 2001, LNCS 2020, pp.266-281.

Ning Huang, born in 1958, received Master's degree in applied mathematics and computer science from Jiangxi University, China in 1991, awarded senior engineer of the Industrial and Commercial Bank of China in 2001. He is now with Center of Modern Educational Technology, Gannan Normal University, Ganzhou, China,as an associate professor. His research interests include information security and digital campus.