Signature-in-Signature Verification Via a Secure Simple Network Protocol

Ming Su, *Member, IEEE*, Min Li, Tieniu Gao, Bin Dong
Department of Computer Science
College of Information Technical Science
Nankai University, Tianjin, China
nksuker@gmail.com

Abstract—**Handwritten signature is often used for verification of important documents. Attackers can fake someone's handwriting signature in a paper-based document, or deliberately attach someone's handwriting signature in binary image to an electronic document, which may print later, claiming the agreement of the signer. In order to solve this problem, we propose a novel signature-in-signature verification system via a secure simple network protocol. A user (client) sends an electronic document to a signer (server) after initial steps of identity confirmation and key negotiation, then the signer embeds some secure information related to the document and the signer into the handwritten signature. Later the signer sends back the encrypted document together with the fragile watermarked signature. Finally the user stores the document with signature, which can be verified by any third person. Our implementation is a simple UDP-based client-server system with some necessary security functions.**

Keywords- signature-in-signature, verification, fragile watermark, binary image, network protocol.

I. Introduction

Handwritten signatures have long been established as the most widespread means of personal verification. Signatures are generally recognized as a legal means of verifying an individual's identity, such as confirmation for a bank payment, and agreement with an important contract, etc. Moreover, verification by signature is comparatively easy and people are familiar and convenient with the use of signatures in their daily life. D. Impedovo and G. Pirlo present the state of the art in automatic signature verification [1]. As for the signature verification system, F. Alonso-Fernandez et al, describe a highly versatile and scalable prototype for Web-based secure access using signature verification. The proposed architecture can be easily extended to work with different kinds of sensors and large-scale databases [2]. A. Julita et al, propose the method of Support Vector Machine (SVM) for verifying the signature in an online signature verification system [3]. J. Trevathan, A. McCabe and W. Read describe a system for enhancing the security of online payments using automated handwritten signature verification, combining complementary statistical models to analyze both the static features of a signature and its dynamic features [4]. Most of these verification systems are based on signal processing and pattern recognition, which have a certain false rate. Nowadays the electronic versions of documents have been widely used in the office and administrative affairs. In the traditional cases people edit and print the electronic document, then sign their names on it as a proof. However, some are very busy and cannot sign the document by themselves. One of the common solutions is that they forward their prepared signatures in digital format, e.g., binary image, to the person who needs the signature. Then the person attaches this signature to the electronic document and prints it out as a signed document. But this method will lead to some serious problems. An attacker can store this format of digital signature, and use it whenever he wants to claim the confirmation of the faked signer for a document. Digital signature is a secure technique avoiding such kind of attacks [5], where an additional code computed by some shared secret and message digest is accompanied with the original document as a proof. But this additional proof still needs some computations for confirmation, which is not as conveniently recognizable as the handwritten signature.

We propose a novel signature-in-signature verification system which accords with the traditional user habit and the basic security requirements. A user sends an electronic file to a signer after initial steps of identity confirmation and key negotiation, then the signer embedding an invisible fragile watermark with some secure information. Afterwards the signer sends back the encrypted electronic file together with the watermarked signature, which can be verified later. The implementation is a simple UDP-based client-server system, in which a simple network protocol is designed for identity confirmation and secure transmission, and topology based fragile watermark for binary image is used for data hiding.

The paper is organized as follows. In Section II, we design a secure transmission protocol for a UDP-based client-server system. In Section III, we briefly introduce the signature-in-signature technique using a data hiding method for binary image. In Section IV, we present the integrated verification system. Some security issues and extensions are discussed in Section V.

II. Secure Transmission Protocol

A. Protocol

The designed transmission protocol includes four stages illustrated in Fig. 1.

1) Identity confirmation for user (Client) and signer (Server);
2) Diffie-Hellman Key Exchange, negotiating a session key;
3) The client uploads an electronic document to be signed;
4) The server transmits the document with a watermarked signature.

In stage 1, the client sends a JOIN_REQ packet to initiate identity confirmation with the server. Then the server responds with a PASS_REQ packet, which is a password request to the user. And then the client sends a PASS_RESP packet to the server including the password. Afterwards the server will verify the password and if the password is correct, the server sends a PASS_ACCEPT packet to the client. If the password is incorrect, the server sends a PASS_REQ packet again to the client. The PASS_REQ packet will be transmitted totally at most three times. In stage 2, the client sends a DH_PAR packet to initiate the key exchange, including the open parameters (a large prime and a primitive root) of the Diffie-Hellman protocol. Then the client sends a DHKEY_ST packet that contains a computed power. Afterwards the server also sends a similar DHKEY_ED packet. Thus the client and server share a common session key, which can be used for symmetric encryption later. Oakley and ISAKMP can be adopted instead of Diffie-Hellman protocol for enhanced security reasons [6, p.201]. In stage 3, the client uploads DOC_DATA packets, which are the document segments encrypted by the session key. Stream Cipher such as RC4 can be used as the encryption algorithm. And the client sends a TERMINATE packet for integrity check purpose, which is the hash digest (such as Sha1) of the entire document. In stage 4, the server recovers the document and checks its integrity. If the check passes, the server will watermark the handwritten signature. He sends the DOC_DATA encrypted packets, and the SIG_DATA packets including the signature with some secure embedding information. Then a TERMINATE packet is transmitted.

We assume that the server handles one client at a time in our implementation. In the case of password errors more than three times, or the disagreement of the signer on the receiving document, the server sends a REJECT packet to the client, where the client closes the session, and the server exits as well. In the case of a packet loss, which will lead to a gap in PacketID numbers, or the unmatched case for the digest of the received file, the system will also terminate.

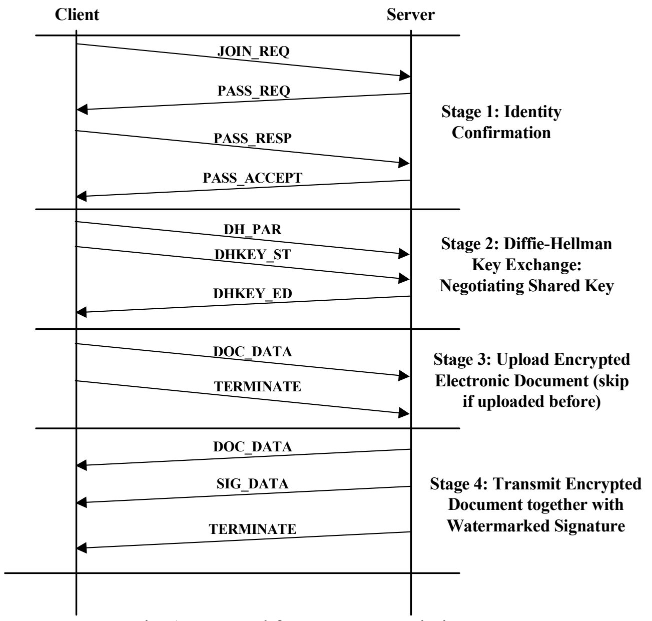
Client
Server
JOIN_REQ
PASS_REQ
PASS_RESP
PASS_ACCEPT
Stage 1: Identity Confirmation
DH_PAR
DHKEY_ST
DHKEY_ED
Stage 2: Diffie-Hellman Key Exchange: Negotiating Shared Key
DOC_DATA
TERMINATE
Stage 3: Upload Encrypted Electronic Document (skip if uploaded before)
DOC_DATA
SIG_DATA
TERMINATE
Stage 4: Transmit Encrypted Document together with Watermarked Signature

Fig. 1. Protocol for secure transmission

B. *Packet Types*

The various packet types mentioned in the protocol are illustrated in Fig. 2. All packets have Header with 1 or 2 bytes. For the JOIN_REQ, PASS_REQ, PASS_ACCEPT, and REJECT packets, there is only a header flag. For the PASS_RESP packet, Payload Length is the number of bytes for the password that the client logs in. For the DH_PAR packet, Payload Length is the number of bytes for the open parameters in the Diffie-Hellman protocol. Besides, for the DHKEY_ST and DHKEY_ED packets, Payload Length is the length of ComputedPower. Recommended key size is 128 bytes for RSA, or 20 bytes for ECC (Elliptic Curve Cryptosystem). And for the TERMINATE packet, Payload Length is the digest length of the used hash function, e.g., 20 bytes for Sha1. Furthermore, for the DOC_DATA and SIG_DATA packets, Payload Length is the length of the data segment transmitting, Packet ID is the sequential number assigned to this packet, and Data is the data segment.

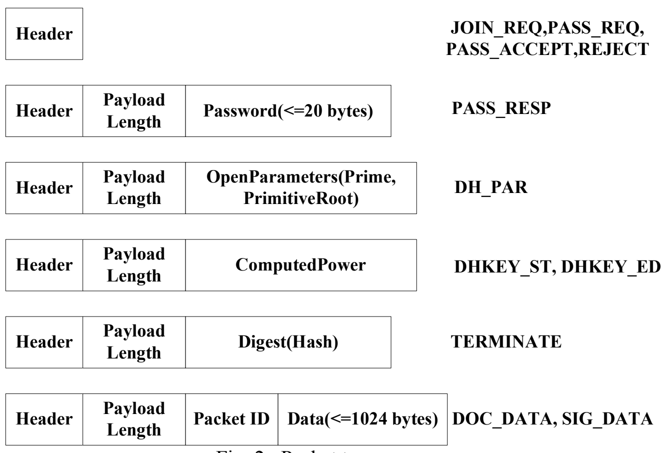

Fig. 2. Packet types

III. SIGNATURE-IN-SIGNATURE USING A FRGILE WATERMARK FOR BINARY IMAGE

Digital Watermarking and data hiding techniques have been proposed for a variety of digital media applications, such as image, audio, and video copyright protection, copy control and authentication. Hiding data in binary image, though difficult, is getting more demands in our daily life. An increasing large number of binary digital images have been widely used, such as black and white wood engraving paintings, handwritten signature, and scanned text etc. Due to the binary image format, most of the relevant hiding methods are based on flipping white or black pixels without causing visible artifacts, doing some complicated computations on lookup table with scores, or some defined transition rules to determine the flipping pixels in priority [7], [8]. This motivates us to explore a new way with simpler calculations determining the flipping pixels. We propose a novel topology based data hiding scheme for binary image [9]. Cutting the original image into equal blocks, and considering the topological structure for the chosen embeddable blocks with almost balanced number of white pixels and black pixels, we set a mapping between the embeddable blocks hiding before and after. We embed one bit via the odd-even enforcement in each embeddable block, and keep the invariant embeddable features in the mapping. The hidden data can be extracted without using the original image as a blind watermark scheme. Theoretical analysis and experimental results show that our scheme is very efficient in the hiding and extraction process, and has satisfied hiding capacity and visual quality.

Handwritten signature is widely used for verification, which can be easily obtained from devices capable of capturing signature signals such as Handwritten Pads, Tablet PCs. Now we adopt this data hiding technique for handwritten signature in binary image format. Illustrated in Fig. 3, we deal with a handwritten signature. The original signature is divided into 3*3 blocks, and we construct a set of embeddable blocks. Based on the set, we embed one bit in each embeddable block via parity enforcement. For instance, for local 3*3 block (a) in the up-left corner, we need to embed '0' in this block. According to the specified invariant topology feature, we transform block (a) into (b) with one pixel flipping. The original signature and the watermarked signature are hard to distinguish, and we have ideal hiding capacity. Our hiding scheme is fragile for integrity authentication, and blind without the original image for information extraction.

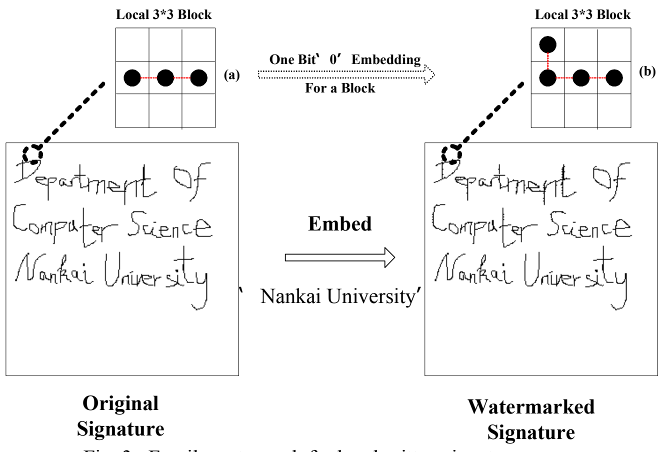

Fig. 3. Fragile watermark for handwritten signature

IV. INTEGRATED SYSTEM

We now introduce the signature-in-signature verification system, combining the secure transmission protocol specified in section II and the fragile watermark technique for handwritten signature described in section III.

Fig. 4 indicates the participants in the integrated system, which includes the following:

1. **User:** In the office environment, a staff prepares some electronic document, which might be a contract, an application, etc. He needs the agreement of his manager in the form of handwritten signature, but the manager is not on the scene.

2. **Signer:** A signer is a person who has the right to approve the document. He will inspect the main content of the document, and then sign it if he agrees.

3. **Verifier:** A verifier is any third person that can check the validity of the signed document. He may glance at the printed-paper version, and will skip further check if he has no doubt. Otherwise he will ask for the electronic version with more investigations, which will be explained detailedly later.

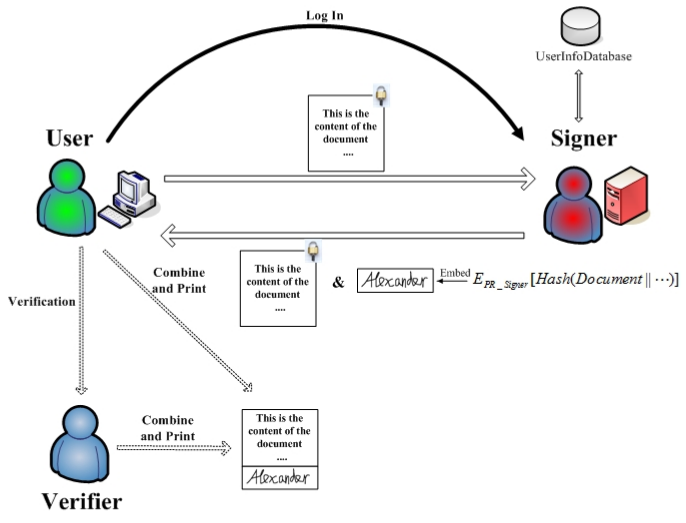
Log In
UserInfoDatabase
User
Signer
This is the content of the document ....
Verification
Combine and Print
This is the content of the document ....
&
Alexander
Embed
$E_{PR\_Signer}[Hash(Document\,||\cdots)]$
Combine and Print
This is the content of the document ....
Alexander
Verifier

Fig. 4. Signature-in-signature verification system

We now briefly describe the sequence of events that happen in the system.

1. **The user logs in the system and the signer verifies his identity.** The user inputs some necessary parameters in the command line, such as the server name (IP address), server port, client port, and password etc. Then the signer will associate the client with the user info database, searching for the stored password. For the matched case of the password, the user successfully logs in.

2. **The user and the signer negotiate a session key by Diffie-Hellman key exchange.** The user chooses global parameters, a large prime q and a primitive root α and sends them to the signer. Then he generates a random number X_A and sends $Y_A = \alpha^{X_A} \bmod q$ to the signer. Similarly the signer generates X_B and sends $Y_B = \alpha^{X_B} \bmod q$ to the user. Thus they have a shared session key $K=(Y_B)^{X_A}=(Y_A)^{X_B}$.

3. **The user sends the encrypted original electronic document.** The user starts the transmission of the document, which will be divided into several segments. For safety reasons, the Data part (illustrated in Fig. 2) of the DOC_DATA packet will be encrypted by the shared session key K.

4. **The signer deals with the handwritten signature, embedding some secure information.** First the signer computes the digest of the decrypted document with some necessary information (time, userinfo) $Hash(Document\,||\cdots)$, then he encrypts it with his private key PR_Signer, and embeds $E_{PR\_Signer}[Hash(Document\,||\cdots)]$ into the handwritten signature binary image, where the recommended public key system is ECC (Elliptic Curve Cryptosystem) due to the short key size(20 bytes or so for guaranteed security).

5. **The signer sends back the encrypted document, together with the watermarked signature.** The document is separated into several packets still encrypted by K, and the watermarked signature is divided into SIG_DATA packets transmitting to the user. Also the TERMINATE packet for the digest of the document and watermarked signature is transmitted for integrity check.

6. **The user stores the electronic version of the document and the watermarked signature, combining them for print.** The user verifies the integrity of the decrypted document and watermarked signature. If the check passes, he will integrate them and print.

7. **The verifier checks the electronic version.** If the verifier is suspicious of the paper-based version, he can check the electronic version. First he extracts the hidden information from the watermarked signature, then he associates the signature with the corresponding public key. For example in Fig. 4, he will search for the public key of 'Alexander'. And then he decrypts the extracted information using this key and gets '$Hash(Document \| \cdots)$'. Finally he compares it with the computed digest at his side. For the matched case, the check passes and he can print it as a receipt.

V. SECURITY CONSIDERATIONS AND SOME EXTENTIONS

Generally there are high risks in the network environment. As for our system, there exist some common threats as follows.

1. The attacker wants to change the content of the document.
2. The attacker wants to fake a valid signature on a document.
3. The attacker wants to change the signature of others.
4. The attacker stores a previous valid signature for later use.

For our system, these threats can be avoided. As for case 1, our design secure transmission protocol ensures the correct Packet ID, and the integrity of the transmitted content. Once there is any transmission error the receiver side can detect it. Besides, the Data part of the packets are encrypted by the session key that is only shared by the user and signer, which prevents the modification on the transmitted content. As for case 2, only the valid user can log in the system and obtain the watermarked signature of the signer. If the attacker gets the handwritten signature in binary format and attaches it to an electronic document, the check will not pass since there is no valid embedding information in the signature. Even if the attacker fakes a handwritten signature on a paper-based document, the verifier can ask for an electronic version for check. Our security methods prevent him faking such version since he does not have the private key of the signer. As for case 3, the attacker can extract the hidden information from the signer, and embed it in another person's signature. However, the verifier will choose a different public key when he sees the handwritten signature, resulting in comparison failure later. As for case 4, the attacker stores a previous watermarked signature, and attaches it for another document. But the embedding information of the previous handwritten signature is related to the previous document, not any other document. Thus the check will not pass.

We implement a UDP-based client-server system for Windows and Linux OS, with the basic features of identity confirmation, session key negotiation, database association, embedding some information into the handwritten signature in binary image, and transmitting files with integrity check via a simplified transmission protocol. There are also some extensions for our system. The multiple users and multiple signers can be supported simultaneously. Besides, the functions of the signer can be further decomposed, i.e., the signature server who is responsible for embedding secure information into handwritten signatures, and the new signer who is responsible for approving a document. The signature server stores the database of the user info and the handwritten signatures of many signers. Thus our system can be extended to large-scaled and more real environment.

ACKNOWLEDGMENT

The corresponding author Dr. Ming Su is supported by the Ph.D. Programs Foundation of Ministry of Education of China(200800551005), the Basic research funding of central Universities of China (07565010211), National Natural Science Foundation of China (Grant No. 61003070); Tianjin basic applied research fund key project(08JCZDJC22100), and Colleges and universiteis science and technology development fund projects in tianjin(20071324). He is also grateful to Ms. Dandan Jia who greatly improved the English.

REFERENCES

[1] D. Impedovo and G. Pirlo, "Automatic signature verification:The state of the art", *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS*, vol. 38, No. 5, pp. 609-635, Sep. 2008

[2] F. Alonso-Fernandez, J.Fierrez-Aguilar, J. Ortega-Garcia, and J.Gonzalez-Rodriguez; "Secure access system using signature verification over tablet PC", *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, Issue 4, pp. 3 – 8, Apr. 2007.

[3] A. Julita, S. Fauziyah, O. Azlina, B. Mardiana, H. Hazura, and A.M. Zahariah, "Online signature verification system", *5th International Colloquium on Signal Processing and Its Application*, 2009 , pp. 8 – 12.

[4] J. Trevathan, A. McCabe and W. Read, "Online payments using handwritten signature verification", *Sixth International Conference on Information Technology: New Generations*, 2009, pp. 901 – 907.

[5] B. Schneier, *Applied Cryptography*, John Wiley & Sons, Inc, 1996.

[6] W. Stallings, *Network Security Eessentials, Applications and Standards*, Third Edition, Pearson Education, 2007.

[7] M. Wu, and B. Liu, "Data hiding in binary images for authentication and annotation", *IEEE Transactions on Multimedia*, vol. 6, No. 4, pp. 528-538, Aug. 2004.

[8] H. Yang and A. C. Kot, "Pattern-based data hiding for binary image authentication by connectivity-preserving", *IEEE Transactions on Multimedia*, Vol.9, No.3, pp. 475-486, Apr. 2007.

[9] T. Gao and Ming Su, "Topology based fragile watermark for binary image", *6th International Conference on Digital Content, Multimedia Technology and its Applications*, Aug. 16-18, Korea, Seoul, pp. 290-295.

[10] C. Yuan, B. B. Zhu, M. Su, X. Wang, S. Li, and Y. Zhong, "Layered access control for MPEG-4 FGS video," *IEEE Int. Conf. Image Processing, Barcelona*, Spain, vol. 1, Sept. 2003, pp. 517-520.

[11] W. Richard Stevens, *UNIX Network Programming*, Prentice Hall, 1990.