Published in *Journal of Computer Security*, Vol. 5, No. 3 (1997), pp. 213–223.

AUTHENTICATION OF SEQUENCES WITH THE SL_2 HASH FUNCTION: APPLICATION TO VIDEO SEQUENCES

Jean-Jacques Quisquater

UCL Crypto Group, Dép. d'Électricité, Université de Louvain
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium

E-mail: jjq@dice.ucl.ac.be

Marc Joye

UCL Crypto Group, Dép. de Mathématique, Université de Louvain
Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium

E-mail: joye@agel.ucl.ac.be

Abstract

This paper presents an interesting application of the Tillich-Zémor function $\mathcal{TZ}$. In particular, we emphasize the concatenation property of this one-way hash function, *i.e.* $\mathcal{TZ}(S \mid T) = \mathcal{TZ}(S)\,\mathcal{TZ}(T)$ where S and T are two binary strings. This property is combined with a multisignature scheme in a journalism context. The aim is to authenticate reports or interviews.

1 Introduction

In this paper, we consider the problem of authenticating a sequence of images that has undergone an editing process that has removed some images. The aim of the authentication is to guarantee that the edited sequence is a subsequence of the original sequence of images. This means for instance that the relative order of the images has not been altered. We also provide an editor with the means needed for his job with only the cryptographic information generated by the camera at the time of the original recording.

Our solution allows to detect any manipulation after the original recording. So, no valid signatures can be generated outside the camera. Therefore, the (secure) camera is committed in the signature process from a secret attached to it. Furthermore, we also commit the person who handles the camera. For this purpose, we use the Guillou-Quisquater multisignature scheme [6].

Hash functions are very sensitive to a modification of even one bit. Hence, we had to take care that the transmission errors would not prevent pieces of editing or

of the original recording from being authenticated. We have considerably reduced the probability of not authenticating an image due to a transmission error (from 63% to 0.77%). This was achieved by dividing the image into smaller entities (blocks).

The paper is organized as follows. In Section 2, we present our model. In Section 3, we review the GQ multisignature scheme for two signing entities. Next, in Section 4, we show how to authenticate images and their relative order. In Section 5, we explain how the editor can efficiently authenticate his work thanks to the Tillich-Zémor hash function [10]. Finally, we conclude in Section 6.

2 Basic functional model

The purpose of our model is to authenticate reports or interviews, because unlike movies the relative order of the images is important. So, we work in the journalism context. The model is basic in the sense that it only includes the minimal amount of "actors" to be functional.

2.1 Functions

Journalist The journalist handles a secure camera to film reports, to make interviews, etc ... He is uniquely identified by a secure object, very often a smart-card [7]. In order to use the secure camera, he has to insert his smart-card. This enables to identify *a posteriori* the journalist who admittedly recorded authenticated images.

Secure camera The camera is *secure*: it holds a secret initialized by an external entity, called trusted third party (TTP). Nobody can get access to this secret. It authenticates each filmed image and the relative order of the images with its own secret and the secret of the plugged smart-card.

TTP The TTP initializes each secure camera with a secret, and delivers smart-cards to journalists. He holds a list of all cameras and smart-cards that he issued.

Editor The editor makes a film by removing some images from the original recording. It is the unique operation he may do; operations such as permutations of images or of sequences are not allowed. He does not produce cryptographic information, he only *transfers* the information needed to authenticate the film and its editing.

Verifier The verifier checks the authenticity of a film. It means that he authenticates each image taken separately and their relative order.

Pirate The pirate plays the role of the cryptanalyst. He tries to break or to weaken the scheme by any means.

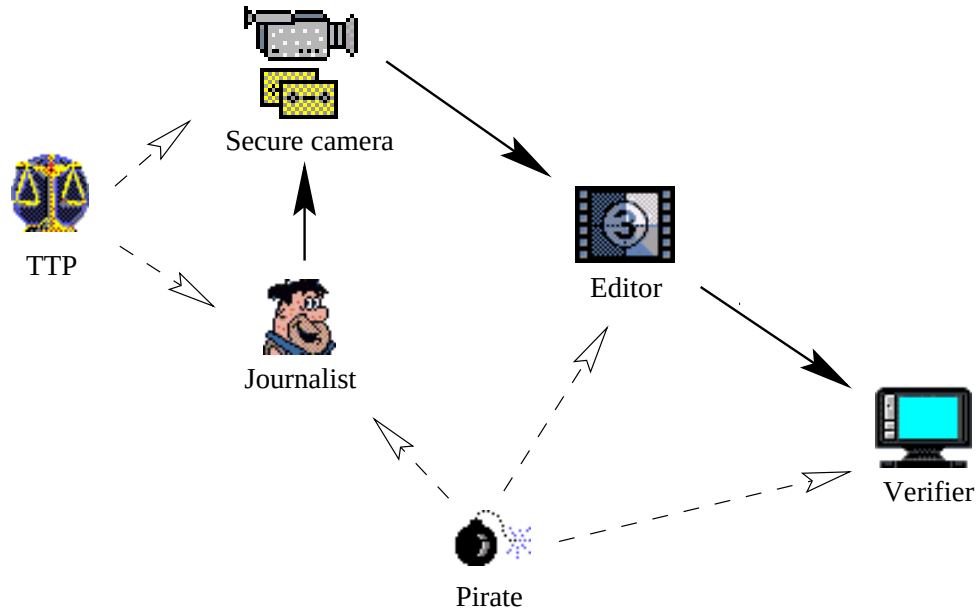

Figure 1 *Functional model.*

2.2 Format of recording

Our scheme applies to digital images using the DV standard [9]. This standard was especially developed for numeric camcorders. It deals with compressed images, dividing by a factor of five the amount of recorded data. So, the resulting video signal is reduced to a bit-rate of approximately 25 Mbps. The compression is intra-frame, similarly to JPEG; there is no temporal compression. Each image is compressed *individually*. This allows to perform a very precise editing.

The DV format includes a sector to write up to 3.5 Mbps of auxiliary data in addition to the video, audio and tracking signals. This is the location where the cryptographic data needed for the authentication will be written.

Finally, the DV format uses errors correcting codes so as to reduce the number of transmission and decoding errors. We will assume that the probability of one bit being incorrectly transmitted or decoded is equal to 10^{-6}.

2.3 Dividing (compressed) images into blocks

At a rate of 25 images per second, an image is coded on $25\,\text{Mbps}/25\,\text{s}^{-1} = 10^6$ bits, on average. The probability to have exactly t errors in an image is given by the Bernouilli formula [8]:

$$\Pr(E = t) = \binom{s}{t} p^t \, (1-p)^{s-t}, \tag{1}$$

where p is the probability to have a transmission error, and s denotes the size of the image. Taking $p = 10^{-6}$, we obtain the following table:

n	s (in bits)	$\Pr(E=0)$	$\Pr(E=1)$	$\Pr(E>0)$	$\Pr(E>1)$
0	1 000 000	0.3679	0.3679	0.6321	0.2642
1	500 000	0.6065	0.3033	0.3935	0.0902
2	250 000	0.7788	0.1947	0.2212	0.0265
3	125 000	0.8825	0.1103	0.1175	0.0072
4	62 500	0.9394	0.0587	0.0606	0.0019
5	31 250	0.9692	0.0303	0.0308	0.0005
6	15 625	0.9845	0.0154	0.0155	$1.2 \cdot 10^{-4}$
7	7 813	0.9922	0.0078	0.0078	$3 \cdot 10^{-5}$
8	3 907	0.9961	0.0039	$4 \cdot 10^{-3}$	$8 \cdot 10^{-6}$

Table 2 *Probability of transmission error for a block of* s *bits.*

The probability to have a least one bit incorrectly transmitted is greater than 63%. Such a probability is not acceptable. In order to overcome this drawback, each (compressed) image i will be divided into 2^n small blocks.

For each block $B_{i,j}$, the camera computes a hash value $h_{i,j} = \mathcal{H}(B_{i,j})$ (with the SHA function [1] for example), and puts it on the auxiliary data sector of the tape. Therefore, if there is one wrong bit for the block $B_{i,j}$, it can be corrected by switching exactly one bit until the corresponding hashing coincides with the hashing $h_{i,j}$ written on the tape. Using this trick, the probability to have a transmission error decreases to the values given in the last column of Table 2. The more the number of blocks is large, the less the transmission error is significant. However, for each image i, the camera has to compute the 2^n hash values $h_{i,j}$, the global hash value I_i

$$I_i = \mathcal{H}(h_{i,1} \mid h_{i,2} \mid \ldots \mid h_{i,2^n}, i), \tag{2}$$

and to write them on the tape. This is time-consuming and the bandwidth for auxiliary data is limited to 3.5 Mbps. A good trade-off is to divide the (compressed)

images into $2^7 = 128$ blocks. That leads to a block size of 7813 bits and a probability of transmission error of $3 \cdot 10^{-5}$. Hence, the probability that a whole image is correctly transmitted is equal to 99.62%.

Remarks. 1) Notice that the computation of I_i in relation (2) includes the number of the image.
2) The cutting of the (compressed) images must be compatible with the DV compression that divides the images into 8×8 blocks on which a DCT is applied. So, the size s in Table 2 is an average value.

3 Guillou-Quisquater multisignature scheme

By signature, we mean a method to prove that the *integrity* of a message has been preserved and that the emitter of the message can be recovered. In order to sign a message, the signer uses his own secret which involves him in the signature process.

In 1988, Guillou and Quisquater [5] developed a zero-knowledge identification scheme minimizing both transmission and memory. This scheme is ideal for smart-card implementations because exchanges with the outside world are time-consuming and the resources of the card are limited. Later, they converted their identification scheme to a signature scheme [6], also suited for smart-card implementations.

Imagine that several entities want to sign the same message. We will illustrate the signature process for two entities, but it may easily be extended to any number of entities. Let's call Carol and Jonathan the signing entities, and Victor the verifying entity. Carol and Jonathan are both characterized by a set of credentials. It consists of identification data that the application warrants, such as bank account number, chip serial number, distinguished name, etc ... This set of credentials is transformed in a publicly known way into an integer $\mathcal{J}$. Moreover, each signing entity has an accreditation β which is the secret solution of

$$\mathcal{J}\beta^{\nu} = 1 \pmod{n}, \tag{3}$$

where the composite integer n and the exponent ν are both public.

Let $\mathcal{J}_C$ (resp. $\mathcal{J}_J$) represent the identity of Carol (resp. Jonathan), and let β_C (resp. β_J) be the secret accreditation of Carol (resp. Jonathan). Let Z_i the message being signed. The public parameters are $\mathcal{J}_C$, $\mathcal{J}_J$, ν and n. Then, the protocol goes as follows.

1. Carol chooses randomly an integer $r_{i,C}$ such that $1 \leq r_{i,C} \leq n-1$, and computes $T_{i,C} = r_{i,C}^{\nu} \pmod{n}$.

2. Similarly, Jonathan computes $T_{i,J} = r_{i,J}^{\nu} \pmod n$.
3. Carol and Jonathan compute $T_i = T_{i,C}\, T_{i,J} \pmod n$ and the challenge $d_i = \mathcal{H}(Z_i, T_i)$, where $\mathcal{H}$ is a one-way hash function.
4. Carol computes $D_{i,C} = r_{i,C}\, \beta_C^{d_i} \pmod n$.
5. Jonathan computes $D_{i,J} = r_{i,J}\, \beta_J^{d_i} \pmod n$.
6. Carol and Jonathan compute $D_i = D_{i,C}\, D_{i,J} \pmod n$.

The signature of message Z_i consists of the pair (d_i, D_i).

To verify that the (d_i, D_i) is a valid signature for Z_i, Victor computes $\mathcal{J} = \mathcal{J}_C\, \mathcal{J}_J \pmod n$ and $T_i{}' = D_i^{\nu}\, \mathcal{J}^{d_i} \pmod n$. Next, he computes $d_i{}' = \mathcal{H}(Z_i, T_i{}')$. If $d_i{}' = d_i$, then Victor accepts the signature.

4 Authenticating images

4.1 Recording

Each (compressed) image i is divided into 128 blocks $B_{i,1}, B_{i,2}, \ldots, B_{i,128}$, and to each block $B_{i,j}$ is associated a hash value $h_{i,j}$ that will be written on the auxiliary data sector of the tape.

By abuse of notations, we will sometimes say image I_i (even though I_i represents the global hash value of the (compressed) image i). A sequence consisting of images r to s will be denoted by Σ_r^s.

Suppose the film consists of k images represented by $\Sigma_1^k = \{I_1, I_2, \ldots, I_k\}$. In order to authenticate the images and their position, a one-way function $\mathcal{H}$ is used recursively. If Z_i denotes the *succession witness* until the i^{th} image, then it is defined by

$$Z_i = \mathcal{H}(Z_{i-1}, I_i) \qquad (1 \leq i \leq k), \tag{4}$$

with $Z_0 = \emptyset$. Each Z_i $(1 \leq i \leq k)$ is signed with a GQ multisignature. The notations are the same as in Section 3, where the camera plays the role of Carol and the journalist plays the role of Jonathan. Call (d_i, D_i) the corresponding GQ multisignature. The succession witness and its signature are written on the auxiliary data sector of the tape.

Remarks. 1) All the data written on the auxiliary data sector use errors correcting codes in order to reduce the probability of bit error to 10^{-6}. Since the size of $h_{i,j}$, Z_i, d_i and D_i are relatively small, the probability to have an error is negligible.

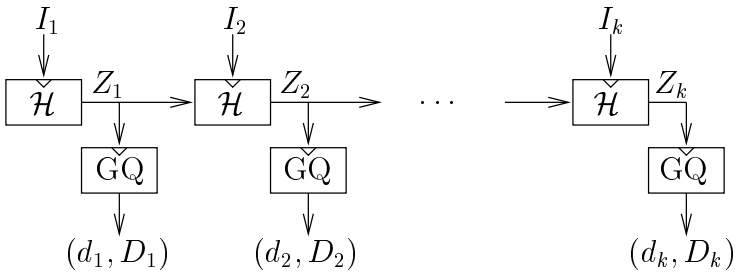

Figure 3 *Recording.*

2) If the hash function $\mathcal{H}$ produces an output of 160 bits (like SHA) and if the GQ multisignature is performed with a 1024-bit long modulus n, then we use less than 17% of the capacity of the auxiliary sector of the tape.

4.2 Editing

The job of the editor is to select some images from the original recording $\Sigma_1^k = \{I_1, I_2, \ldots, I_k\}$ while respecting the order. Moreover, he has to ensure the later authentication of the edited film. So, if a sequence $\Sigma_\ell^{\ell+m}$ is removed, then he has to furnish $Z_{\ell+m}$ in order to reinitialize the succession witness computation for the next sequence. Actually, the verifier needs $Z_{\ell+m}$ to compute $Z_{\ell+m+1}, Z_{\ell+m+2}, \ldots$

After editing, the final film consists of a succession of sequences Σ_p^q with $q \geq p$. Note that an isolated image corresponds to the case $p = q$.
For each sequence Σ_p^q that he keeps, the editor writes on the final tape

(a) the (compressed) images making up the sequence and their number, *i.e.* i and $B_{i,j}$, for $p \leq i \leq q$ and $1 \leq j \leq 128$;

(b) the hash values of the 128 blocks of images p to q, *i.e.* $h_{i,j}$, for $p \leq i \leq q$ and $1 \leq j \leq 128$;

(c) the GQ signature of the succession witness until the q^{th} image, *i.e.* (d_q, D_q).

The hash values $h_{i,j}$ are given in order to reduce the probability of transmission errors (see Paragraph 2.3). However, there is another advantage: they allow the editor to modify one or several blocks inside an image *without* "des-authenticating" the whole image as we will see in the verification process. This enables, for example, to insert a logo or to mask a part of an image.

4.3 Verification

Let Σ_1^k be the original recording, and let $\left\{\Sigma_{p_1}^{q_1}, \Sigma_{p_2}^{q_2}, \ldots, \Sigma_{p_u}^{q_u}\right\}$ be the final film after editing. In order to authenticate the film, the verifier has to check each image separately and their order as follows.

First, he computes the hash value of each block $B_{i,j}$ making up the final film: $h'_{i,j} = \mathcal{H}(B_{i,j})$. After a possible modification of one bit, if $h'_{i,j} \neq h_{i,j}$ (where $h_{i,j}$ is read on the tape), then the block $B_{i,j}$ has been modified. Also, he computes the global hash value of each image i making up the final film: $I'_i = \mathcal{H}(h_{i,1} \mid \ldots \mid h_{i,128}, i)$. Notice that the computation of I'_i is achieved using $h_{i,j}$ read on the tape and not the computed $h'_{i,j}$.

Next, for each sequence $\Sigma_{p_j}^{q_j}$, he computes recursively the succession witness Z_{q_j} by

$$Z_m = \mathcal{H}(Z_{m-1}, I_m) \quad \text{for } p_j < m \leq q_j, \tag{5}$$

where the initial value Z_{p_j} is read on tape.

Then, he checks that the GQ signatures (d_{q_j}, D_{q_j}) of each succession witness Z_{q_j} $(1 \leq j \leq u)$ are valid (see Section 3). Because the one-wayness of the hash function $\mathcal{H}$, it results that all the images (containing *unmodified* blocks) of a *given sequence* $\Sigma_{p_j}^{q_j}$ must be authentic. Otherwise, only the part of the image containing the unmodified blocks will be authenticated. It is not possible to find other images for the sequence $\Sigma_{p_j}^{q_j}$ that will give the final succession witness Z_{q_j} (which is signed), because this problem is equivalent to find collisions for the hash function.

Finally, to verify the relative order of the sequences, the verifier has to check that $p_j > q_{j-1}$ for $2 \leq j \leq u$. It is not possible to modify the image number i because, by Eq. (2), this number is tied to the content of the image when I_i is computed.

5 Identifying the editor

From an edited film, anybody can produce another authenticated film by removing some images for example. So, if the editor wants to prevent such manipulations, he has to protect his work by signing it. Moreover, the signature process enables to identify the editor.

For this purpose, the editor can use a method similar to the one developed in Section 4. But in that case, he has to compute a lot of hash values. Another possibility is to sign individually each sequence $\Sigma_{p_j}^{q_j}$. These solutions are not ideal, we need another tool.

5.1 Tillich-Zémor hash function

Let $\mathrm{SL}_2(\mathbb{F}_{2^r})$ be the group of two-dimensional unimodular matrices with entries in the quotient field $\mathbb{F}_{2^r} = \mathbb{F}_2[X]/P_r(X)$, where $P_r(X)$ is an irreducible polynomial (over $\mathbb{F}_2$) of degree r. Define the matrices $A, B \in \mathrm{SL}_2(\mathbb{F}_{2^r})$ by

$$A = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}.$$

Let the mapping $\pi : \{0,1\} \to \{A, B\}, \begin{cases} 0 \mapsto A \\ 1 \mapsto B \end{cases}$.
Then, the hashcode of the binary string $S = (s_1, s_2, \ldots, s_k)_2$ is the matrix product

$$\mathcal{TZ}(S) = \pi(s_1)\pi(s_2)\cdots\pi(s_k). \tag{6}$$

Proposition 1. $< A, B > = \mathrm{SL}_2(\mathbb{F}_{2^r})$*, i.e. the subgroup generated by* A *and* B *is the whole group* $\mathrm{SL}_2(\mathbb{F}_{2^r})$. □

Therefore, if r is sufficiently large, then probabilistic attacks such as the birthday paradox do not apply, since the set of hashcodes has $|\mathrm{SL}_2(\mathbb{F}_{2^r})| = 2^r(2^r - 1)(2^r + 1)$ elements. Tillich and Zémor [10] recommend to choose r in the range $130 - 170$.

Proposition 2 (Concatenation property). *Let* R *and* S *be two binary strings. If "* $|$*" denotes the concatenation symbol, then*

$$\mathcal{TZ}(R \mid S) = \mathcal{TZ}(R)\,\mathcal{TZ}(S). \tag{7}$$

□

For security purposes, the polynomial $P_r(X)$ must be carefully chosen. The authors of [2] (see also [3]) showed that if the order of A respectively to B is relatively small, then clashing binary strings may be found. However, this attack fails if the factorization of both $2^r - 1$ and $2^r + 1$ contains large primes.

On the other hand, Geiselmann [4] found a technique for finding clashing strings by embedding A and B into finite fields. With his method, he reduces the problem of finding clashing sequences to the problem of computing discrete logarithms in $\mathbb{F}_{2^r}$. However, these clashing strings have very strange structures (very long sequences of zeros and ones) and can thus not really be considered as an attack.

5.2 Authenticating images (second version)

The second scheme is exactly the same as described in Section 4, except that the succession witness until image i is now defined by

$$Z_i = Z_{i-1}\,\mathcal{TZ}(I_i) \qquad (1 \leq i \leq k), \tag{4'}$$

with $Z_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

The main advantage of the SL_2 hash function is the concatenation property. As before, let $\left\{\Sigma_{p_1}^{q_1}, \Sigma_{p_2}^{q_2}, \ldots, \Sigma_{p_u}^{q_u}\right\}$ be the film after editing. From Proposition 2, we have

$$\begin{aligned} &\mathcal{TZ}(I_{p_1} \mid \ldots \mid I_{q_1} \mid I_{p_2} \mid \ldots \mid I_{q_2} \mid \ldots \mid I_{p_u} \mid \ldots \mid I_{q_u}) \\ &\quad = \mathcal{TZ}(I_{p_1} \mid \ldots \mid I_{q_1})\,\mathcal{TZ}(I_{p_2} \mid \ldots \mid I_{q_2}) \ldots \mathcal{TZ}(I_{p_u} \mid \ldots \mid I_{q_u}) \\ &\quad = Z_{p_1-1}^{-1} Z_{q_1} Z_{p_2-1}^{-1} Z_{q_2} \ldots Z_{p_u-1}^{-1} Z_{q_u}. \end{aligned} \tag{8}$$

Proof. Obvious, since

$$Z_{q_j} = \mathcal{TZ}(I_1 \mid \ldots \mid I_{p_j-1} \mid I_{p_j} \mid \ldots \mid I_{q_j}) = Z_{p_j-1}\,\mathcal{TZ}(I_{p_j} \mid \ldots \mid I_{q_j}).$$

□

Therefore, the editor has just to compute the matrix product

$$\widetilde{Z} = Z_{p_1-1}^{-1} Z_{q_1} Z_{p_2-1}^{-1} Z_{q_2} \ldots Z_{p_u-1}^{-1} Z_{q_u}, \tag{9}$$

and signs it with the GQ scheme.

To authenticate the editing, the verifier computes $\widetilde{Z}$ where the values of Z_{p_J-1} are read on the tape and checks the validity of the signature.

Remark. If the editor does not trust the camera, he can re-iterate the whole process of verification described in Paragraph 4.3. If he only wants to ensure that what he signs corresponds to filmed images, then he may check the signatures of the $Z_{p_j} - 1$ and Z_{q_j} used in the computation of $\widetilde{Z}$.

5.3 Further applications

The SL_2 function is interesting when several hash values have to be used several times. We have seen a first application for a secure camera in the journalism context. Another application may be the following.

Imagine that a secure camera takes photographs. Each slide S_i is individually hashed into $Z_i = \mathcal{H}(S_i)$. Then, the hash values are signed by the secure camera.

Let $\{S_1, \ldots, S_k\}$ be the original set of slides. If an editor selects some slides and alters their relative order, he can efficiently sign his work if the function $\mathcal{H}$ is the Tillich-Zémor hash function. Indeed, if $\{S_{r_1}, S_{r_2}, S_{r_\ell}\}$ denotes the set of slides resulting for the editing, then he just has to compute $\widetilde{Z} = \prod_{j=1}^{\ell} Z(I_{r_j})$ and to sign it.

This scheme becomes really interesting in the case of an editor using a (trusted) database of slides with precomputed hash values: the editor is able to authenticate several editings in a very efficient way.

6 Concluding remarks

A system of authenticating images and their relative order was developed, even if an editor has removed some images from the original recording. The main advantage of our system is that only *one* signature is needed to authenticate a whole sequence of images. Indeed, if an obvious solution (such as signing each image to which a number is tied) is used, the verifier has to check k signatures for a sequence of k images. So, we reduced the problem of authenticating of sequence of k images to the problem of verifying only one signature.

Moreover, with only one signature per sequence, we can authenticate the order of the sequences. Also, our model allows the editor to modify one or several blocks of an image, but can also authenticate the remainings of the image.

Finally, we show that the Tillich-Zémor hash function enables the editor to minimize the computations to authenticate his work. He just has to perform some matrix products and to sign the resulting value.

Note that our scheme does not take into account the issue of synchronization of the sound and the related authentication problems. This will be done in a future work.

Acknowledgments We are grateful to Jean-François Delaigle for some fruitful discussions. We are also grateful to an anonymous referee for providing some useful comments in order to enhance the quality of this paper.

References

[1] FIPS 180-1. *Secure Hash Standard*. NIST, US Department of Commerce, Wahington D.C., Apr. 1995.

[2] CHARNES, C., and PIEPRZYK, J. Attacking the SL_2 hashing scheme. In *Advances in Cryptology – ASIACRYPT '94* (1995), J. Pierprzyk and R. Safavi-

Naini, Eds., vol. 917 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 322–330.

[3] CHARNES, C., and PIEPRZYK, J. Weak parameters for the SL_2 hash function. Preprint, 1996.

[4] GEISELMANN, W. A note on the hash function of Tillich and Zémor. In *Cryptography and Coding* (1995), C. Boyd, Ed., vol. 1025 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 257–263.

[5] GUILLOU, L. C., and QUISQUATER, J.-J. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology – EUROCRYPT '88* (1988), C. G. Günther, Ed., vol. 330 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 123–128.

[6] GUILLOU, L. C., and QUISQUATER, J.-J. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology – CRYPTO '88* (1990), S. Goldwasser, Ed., vol. 403 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 216–231.

[7] GUILLOU, L. C., UGON, M., and QUISQUATER, J.-J. The smart card: A standardized security device dedicated to public cryptology. In *Contemporary cryptology - The science of information integrity* (1992), G. J. Simmons, Ed., IEEE Press, pp. 561–613.

[8] SHIRYAYEV, A. N. *Probability*, vol. 95 of *Graduate Texts in Mathematics*. Springer-Verlag, 1984.

[9] SONY. The digital video handbook, 1995.

[10] TILLICH, J.-P., and ZÉMOR, G. Hashing with SL_2. In *Advances in Cryptology – CRYPTO '94* (1994), Y. Desmedt, Ed., vol. 839 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 40–49.