Watermarking in the Encrypted Domain

Jordi Herrera-Joancomartí[†], David Megías[‡]

[†] Departament d'Enginyeria de la Informació i les Comunicacions
Universitat Autònoma de Barcelona
[‡]Estudis d'Informàtica, Multimèdia i Telecomnicacions
Universitat Oberta de Catalunya
jherrera@deic.uab.cat, dmegias@uoc.edu

Abstract. In this paper we discuss the possibilities of insertion and detection of watermarks in the encrypted domain. We review the literature of computing in the encrypted domain and we discuss the application of such techniques in a watermarking scenario to obtain secure protocols for zero-knowledge detection and asymmetric fingerprinting schemes.

Keywords Watermarking, fingerprinting, copyright protection,encryption.

1 Introduction

Encryption functions are those that transform a message (or in general an object, an image or a file) into a piece of nonsense information $I_k = \mathcal{E}_k(I)$. Usually, the only goal of such encrypted information I_k is to allow to transfer or store original data so as to ensure that the original information cannot be derived from its encrypted version. However, what will be the result if the encrypted data is modified? In other words, can we operate in the encrypted domain? The answer depends on which operations we want to perform and if we want to transform such operations back in the clear domain. Obviously, some general transformations can be performed on encrypted data providing the transformation itself has a "meaning" in the encrypted domain. For instance, encrypted data can be compressed [1] in order to save space. Here the compression function, $\mathcal{C}$, has a complete meaning (to save space) in the encrypted domain. Such transformation can be performed on encrypted data provided the decompression function is applied before the decryption process, that is

$$\mathcal{D}_{k^{-1}}(\mathcal{C}^{-1}(\mathcal{C}(\mathcal{E}_k(I)))) = I$$

But if we want to compress the encrypted data in order to obtain a compressed data that can be decrypted to obtain a compressed version of the clear information, then the involved functions must hold the following equation:

$$\mathcal{C}^{-1}(\mathcal{D}_{k^{-1}}(\mathcal{C}(\mathcal{E}_k(I)))) = \mathcal{D}_{k^{-1}}(\mathcal{C}^{-1}(\mathcal{C}(\mathcal{E}_k(I))))$$

[1] Although cryptosystems are normally applied on already compressed data for masking the statistical properties of the clear text, such example may help to illustrate an hypothetic scenario.

and, of course, such condition does not hold for all possible functions. From now on, we will refer to *computing in the encrypted domain* only to those operations that "survive" the decryption process and can be inverted after that.

In general, computing in the encrypted domain could be performed depending on the cryptosystem used and also depending the operations performed in the encrypted domain.

On the other hand, the interest on computing in the encrypted domain is based on the fact that original data can be "meaningfully" modified by anyone without having access to the original information. As we will show in next sections, such property is very useful in some secure protocols.

2 Homomorphic Encryption

A first approach towards computing in the encrypted domain is to find out cryptosystems that allow some basic algebraic operations in the encrypted domain.

Privacy homomorphisms were first proposed by Rivest *et alter* in [1]. Privacy homomorphisms were defined as encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands. In this seminal paper, five simple examples of privacy homomorphisms were presented, the most relevant one was the RSA, which is multiplicative, *i.e.*, it allows multiplication on encrypted data:

$$\mathcal{E}(a \cdot b) = (a \cdot b)^e \bmod n = (a^e \bmod n) \cdot (b^e \bmod n) = \mathcal{E}(a) \cdot \mathcal{E}(b)$$

In [2] an additive and multiplicative privacy homomorphism was presented. The proposal allows addition, subtraction and multiplication to be carried out directly on encrypted data, but it does not allow computation of multiplicative inverses; in other words, it is a ring privacy homomorphism, but not a field privacy homomorphism.

Another well known privacy homomorphism is the one proposed by Paillier [3] used as a building block of different applications.

Security of privacy homomorphisms has also been studied. It is known that if a privacy homomorphism preserves order when encrypting, it is insecure against a ciphertext-only attack; if it is additive, it is insecure against a chosen-cleartext attack [4]. With the exception of RSA, all the examples given in [1] were subsequently broken in [5] using ciphertext-only or known-cleartext attacks. On the other hand, the privacy homomorphism presented in [2] is secure against a known-cleartext attack.

2.1 Homomorphic encryption applications

As we have pointed out, computing in the encrypted domain can be applied in some secure protocols. In particular, homomorphic encryption has been used successfully in different environments as a building block.

For instance, in [6] it is shown that some security properties of a secret sharing scheme can be proven using the homomorphic properties of the secret sharing function.

On the other side, as an extension of a secret sharing, multiparty computation schemes [7] and verifiable signature sharing [8] also use homomorphic encryption as a building block.

Regarding more specific applications, electronic voting systems also use homomorphic encryption. In this case, some properties of homomorphic encryption are used to verify the tally of the ballots without revealing what those ballots are, since homomorphic encryptions allow to have the sum of a group of encrypted values verified without revealing those encrypted values (see [9] for an exhaustive bibliography on voting schemes using homomorphic encryption).

On the other hand, electronic auctions also use homomorphic encryption [10]. For instance, in [11] a sealed-bid auction using homomorphic encryption is proposed and in [12] a proposed sealed-bid auction is based on circuit evaluation using homomorphic encryption.

Homomorphic encryption is also used in the context of databases [13–15] in order to allow database computation while maintaining a certain level of privacy of the records.

In mobile agent technology, homomorphic encryption is also applied. For instance, a partial solution for the problem of protecting agents against malicious hosts is proposed in [16] based on computing with encrypted functions, an extension of computing with encrypted data.

Finally, in [17], the authors take advantage of the homomorphic properties of the RSA to define a publicly verifiable watermarking scheme in which the verification process does not reveal the embedded mark.

3 Cryptographic computation on encrypted data

In addition to the basic algebraic operations described in the previous section, more complex computation can be performed on encrypted data. In particular, encryption and digital signature can be performed.

3.1 Encryption of encrypted data

Recalling again the idea of computing on encrypted domain described in Section 1 in the sense that operations in the encrypted domain can be inverted in the clear domain, cryptosystems which allow encryption in the encrypted domain fall into the category of commutative cryptosystems when both encryption functions are the same.

A commutative encryption function $\mathcal{E}_k(\cdot)$, with a secret key k, has the property that:

$$\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(m)) = \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(m))$$

for all keys k_1,k_2.

Obviously, most symmetric encryption functions are not commutative while RSA is a commutative encryption scheme, since

$$\mathcal{E}_{k_1}(\mathcal{E}_{k_2}(m)) = \mathcal{E}_{k_1}(m^{k_2} \bmod n) = m^{k_2 \cdot k_1} \bmod n = m^{k_1 \cdot k_2} \bmod n = \mathcal{E}_{k_2}(\mathcal{E}_{k_1}(m))$$

Commutative encryption schemes are useful in different areas. For instance, a fair coin flipping protocol and a bit commitment scheme can be defined using commutative encryption [18].

3.2 Digital signature of encrypted data

The interest of digitally sign encrypted data has been discussed in the literature. In [19, 20] the authors argue that it is there is no sense to sign encrypted data since non-repudiation cannot be ensured because the signer does not know what he/she signs. However, at it is pointed out in [21], there are some applications where privacy is more important than non-repudiation.

Blind signatures: a particular approach. The concept of blind signature defined by Chaum in [22] can be seen as an special case of digital signature on encrypted data. In a blind signature scheme, the signer signs some data without knowing which data he has signed. In this case it is assumed that the encrypted data will be always signed and then decrypted before the signature validation, that is given an encryption function $\mathcal{E}$, a decryption function $\mathcal{D}$, a signature function $\mathcal{S}$ and a signature verification function $\mathcal{V}$, the only possible function composition will be

$$\mathcal{V}(\mathcal{D}(\mathcal{S}(\mathcal{E}(m))))) = m$$

In this especial case, $\mathcal{E}$ is a blinding function rather than an encryption function, since the composition with the "decryption" or unblinding counterpart holds $\mathcal{D}(\mathcal{E}(m)) \neq m$.

Assuming such principles, the blinding function can be defined as

$$\mathcal{E}(m) = m \cdot r^e \bmod n = c$$

where $k = r^e$ is the private key with $r \in_R \mathbb{Z}_n$ and e the public key of the signature function $\mathcal{S}$ defined by

$$\mathcal{S}(c) = c^d \bmod n = m^d \cdot r^{ed} \bmod n = m^d \cdot r \bmod n = c'$$

since $d \cdot e \equiv 1 \bmod \phi(n)$.

The unblinding function is defined by

$$\mathcal{D}(c') = c' \cdot r^{-1} = m^d = s$$

and the signature verification function is

$$\mathcal{V}(s) = s^e \bmod n = (m^d)^e \bmod n = m$$

With these definitions, it is possible to sign in the blinded domain and verify in the clear domain, that is

$$\mathcal{V}(\mathcal{D}(\mathcal{S}(\mathcal{E}(m)))) = m.$$

The first application of blind signatures was an anonymous payment system [23] proposed also by D. Chaum.

4 Watermarking in the encrypted domain

In the previous sections, we have presented different approaches to compute in the encrypted domain. In this section, we want to pay attention to the possibility of watermarking in the encrypted domain, that is to say, we want to study when or how encryption and decryption functions can commute with the watermarking operations (typically, mark and verify).

As we mention above, computing in the encrypted domain allows the generation of secure protocols to solve new problems. In the same way, watermarking over encrypted data may produce secure protocols to solve problems like publicly verifiable watermarking schemes or asymmetric fingerprinting schemes.

A watermarking scheme can be defined with two functions, the embedding or marking function $\mathcal{M}$ and the extraction or verification function $\mathcal{V}$. Roughly speaking, the marking function takes an image[2] I as an input together with the mark m to be embedded and produces the marked image $I^* = \mathcal{M}(I, m)$. On the other side, the verification function takes the marked (and possibly distorted) image $\widehat{I}$ and gives the mark embedded in the image $\mathcal{V}(\widehat{I}) = m$.

Again, we denote the encryption function $\mathcal{E}$ with a encrypting key k that produce a encrypted image $I_k = \mathcal{E}_k(I)$. The decryption function $\mathcal{D}$ produces the clear image $I = \mathcal{D}_{k^{-1}}(I_k)$ given the decrypting key k^{-1} and the encrypted image I_k.

Depending on where each function is applied different properties can be satisfied:

Prop. 1 The marking function $\mathcal{M}$ can be executed in an encrypted image I_k to embed a mark m.

Prop. 2 The verification function $\mathcal{V}$ can be able to reconstruct a mark in the encrypted domain when it has been embedded in the encrypted domain.

Prop. 3 The verification function $\mathcal{V}$ can be able to reconstruct a mark in the encrypted domain when it has been embedded in the clear domain.

Prop. 4 The decryption function does not affect the mark integrity in terms of mark reconstruction process.

Property 1 ensures that when the marking function is performed over an encrypted image, the result $I_k^* = \mathcal{M}(I_k, m)$ will have some sense. This point is

[2] For simplicity, we speak about images. However main ideas of the discussion proposed can be applied to digital objects different than images, such as audio files.

relevant since the marking schemes in the literature are based on image characteristics that may disappear when the image is encrypted.

The second property ensures that the mark and verification process can be performed entirely in the encrypted domain

$$\mathcal{V}(\mathcal{M}(\mathcal{E}_k(I), m)) = m.$$

The third property means that the encryption function does not affect the mark integrity in terms of mark reconstruction process and it holds if

$$\mathcal{D}_{k^{-1}}(\mathcal{V}(\mathcal{E}_k(\mathcal{M}(I, m))) = m.$$

Notice that properties 2 and 3 are equivalent in case the marking function and the encryption function commute

$$\mathcal{M}(\mathcal{E}_k(I), m) = \mathcal{E}_k(\mathcal{M}(I, m)) = I_k^*.$$

The last property implies that

$$\mathcal{V}(\mathcal{D}(\mathcal{M}(\mathcal{E}_k(I), m))) = m.$$

At first glance, it could be difficult to determine encryption/decryption functions and watermarking algorithms that hold all the proposed properties. For instance, as it has been pointed out, Property 1 fails for the vast majority of marking schemes in the literature since to achieve imperceptibility they are based on image characteristics that disappear when the image is encrypted.

However, in some scenarios, an interesting secure protocol can be defined only with one of the proposed properties. A clear example is the verification protocol proposed in [17] that allows to demonstrate the presence of a watermarking in an image without revealing the mark, thus producing the first approach to a zero-knowledge watermarking verification protocol. In this case, the watermarking verification protocol proposed is based on any linear and additive watermarking algorithm in which the watermarking verification can be performed by mark correlation (for instance the spread spectrum technique proposed by Cox et al. [24]). The encryption algorithm used is the RSA. When using such watermarking techniques together with this encryption algorithm Property 3 holds due to the homomorphic properties of the RSA algorithm together with the multiplicative operations performed in the verification process. This example shows the possibilities of watermarking in the encrypted domain, even if only one of the properties holds.

If we assume functions $\mathcal{M}$, $\mathcal{V}$, $\mathcal{E}$, $\mathcal{D}$ hold the properties stated above we can define new protocols with some interesting properties as it is shown in the next subsection.

4.1 An asymmetric fingerprinting scheme

In contrast to watermarking (which allows ownership protection), fingerprinting is a technique which allows to track redistributors of electronic information. In a fingerprinting scheme, each copy image is marked with a different mark that allows buyer identification. Usually, it is assumed that two or more dishonest buyers can only locate and delete marks by comparing their copies (marking assumption, [25]).

Classical fingerprinting schemes [26, 25] are symmetrical in the sense that both the seller and the buyer know the fingerprinted copy. Even if the seller succeeds in identifying a dishonest buyer, her previous knowledge of the fingerprinted copies prevents her from using them as a proof of redistribution in front of third parties. In [27], the concept of asymmetric fingerprinting was introduced, whereby only the buyer knows the fingerprinted copy. Different asymmetric fingerprinting proposals can be found in the literature [27–33].

In [29, 27] the authors use multiparty computation [34] to ensure that the seller cannot obtain knowledge about the marked copy. Since multiparty computation is computationally complex, some proposals are focused on avoiding such a technique. For instance, in [30] the cryptographic tool Committed Oblivious Transfer [35] is used instead of multiparty computation. Such scheme has been improved in [32] and [33].

Watermarking in the encrypted domain can offer new possibilities towards the construction of new asymmetric fingerprinting schemes. As an example of a possible application, consider the following scenario.

Suppose functions $\mathcal{M}$, $\mathcal{V}$, $\mathcal{E}$, $\mathcal{D}$ hold the four properties stated above. Suppose the watermarking scheme used is asymmetric in the sense that a different secret information is needed for embedding and verifying the mark [36]. Finally, suppose the function $\mathcal{E}$ is a commutative encryption function in the sense described in Section 3.

With these assumptions, we can define an asymmetric fingerprinting scheme based on a protocol between the seller, S, and the buyer, B that allows to embed a mark into an image in a way that:

– only B obtains the marked image,
– only S knows the original image.

Protocol 1 (The marking protocol)

1. *S, encrypts the original image I using an encryption algorithm $\mathcal{E}$ and a secret key k_1. $I_{k_1} = \mathcal{E}_{k_1}(I)$. Then S sends the encrypted image I_{k_1} and the mark m to be embedded to B.*
2. *B embeds the mark m into the encrypted image I_{k_1} using the marking function $\mathcal{M}$. The marked image is denoted by $I^*_{k_1} = \mathcal{M}(I_{k_1}, m)$. Then B encrypts $I^*_{k_1}$ using a key k_2 and sends the result $I^*_{k_1k_2} = \mathcal{E}_{k_2}(I^*_{k_1})$ to S.*
3. *S verifies that B has embedded the mark, that is, using the verification function $\mathcal{V}$, S extracts the mark m and checks that $\mathcal{V}(I^*_{k_1k_2}) = m$.*
4. *S removes the first encryption with k_1^{-1} over the image $\mathcal{D}_{k_1^{-1}}(I^*_{k_1k_2}) = I^*_{k_2}$ and sends the result to B.*
5. *B removes the second encryption with k_2^{-1} over the image $\mathcal{D}_{k_2^{-1}}(I^*_{k_2}) = I^*$ and obtains the marked image.*

Notice that although B is the one that marks the image, he does not obtain the original image since he only obtains the encrypted version I_{k_1} and the marked one I^*. On the other hand, the scheme proposed is asymmetric (in the sense of fingerprinting) since S does not obtain the marked image I^*.

The correctness of the protocol is based on the assumptions stated before. Step 2 uses Property 1: the marking function can be executed in the encrypted domain. Correctness of Step 3 is based on Property 3: the encryption function does not affect the mark integrity in terms of mark reconstruction. Step 4 is based on the assumption that the encryption function is a commutative encryption function and also on Property 4: the decryption function does not affect the mark integrity in terms of mark reconstruction. Finally, Step 5 is also based on Property 4. On the other hand, the asymmetry assumed for the watermarking scheme allows B to embed the mark in a way that S should not be able to reproduce, although S can verify that the mark is embedded into the marked image. Otherwise, the knowledge of the original image and the mark would allow S to obtain the marked image.

Of course, this is a high level overview of the protocol and we have made many assumptions on the properties of functions $\mathcal{M}$, $\mathcal{V}$, $\mathcal{E}$, $\mathcal{D}$. However, this protocol illustrates how watermarking in the encrypted domain can be applied to obtain new protocols with new properties.

5 Conclusions and further research

In this paper we have discussed the possibilities that watermarking in the encrypted domain offers. We have reviewed the most relevant literature that deals with the problem of computing in the encrypted domain. We have given some examples of privacy homomorphisms which preserve some algebraic operations and other cryptosystems that allow cryptographic operations on encrypted data. We have introduced the concept of watermarking in the encrypted domain. We have defined the four main properties needed in the interaction of the main functions encryption/decryption and marking/verifying. We have shown that some properties have been already used to obtain an approach to a zero-knowledge watermarking verification protocol where the verifier does not obtain the embedded mark after the verification process. Finally we have outlined a high level asymmetric fingerprinting scheme based on watermarking in the encrypted domain.

5.1 Further research

Further research should be directed to obtain functions $\mathcal{M}$, $\mathcal{V}$, $\mathcal{E}$, $\mathcal{D}$ with the aforementioned properties. Different strategies can be used to reach this objective. First of all, a deep study of the existing watermarking schemes and cryptosystems must be carried out in order to determine if functions with those properties already exist. A first approach has shown that Property 3 holds for some specific schemes [17]. Secondly, new watermarking schemes and ad-hoc cryptosystems can be proposed. Here we mean by ad-hoc cryptosystem, a encryption function that encrypts an object producing a same object type as encrypted output (for instance, a clear image that is encrypted giving an encrypted image). Such ad-hoc encryption for images has been studied in the literature [37–40] but the objective of such work has been sometimes undervalued since detractors argue that an image can be considered as a file and can be encrypted using any standard cryptosystem. However, if you need to compute in the encrypted domain you may need the encrypted object be an image and then ad-hoc cryptosystems may have a new development field.

On the other hand, combined watermarking/cryptosystem functions can be defined in order to obtain some of the needed properties. For instance, different functions can be defined providing a function composition holds

$$\mathcal{V}(\mathcal{D}(\mathcal{M}(\mathcal{E}_k(I), m))) = m$$

In this case, the “encryption” function may not be the inverse of the ”decryption” counterpart but this is not an issue if the application order of functions are determined a priori (see for instance, blind signatures -Section 3-).

Furthermore, if such functions can be defined, a detailed study about their security must be carried out before they can be used within a secure protocol.

Acknowledgements

This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 ”ARES”, funded by the Spanish Ministry of Science and Education.

References

1. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In Foundations of Secure Computation, New-York, Academic Press (1978) 169–179
2. Domingo-Ferrer, J.: A provably secure additive and multiplicative privacy homomorphism. In Information Security, Berlin, Springer-Verlag (2002) 471–483 Lecture Notes in Computer Science Volume 2433.
3. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In Cryptology - Eurocrypt ’99, Berlin, Springer-Verlag (1999) 223–238 Lecture Notes in Computer Science Volume 1592.
4. Ahituv, N., Lapid, Y., Neumann, S.: Processing encrypted data. Commun. ACM **30** (1987) 777–780
5. Brickell, E.F., Yacobi, Y.: On privacy homomorphisms. In Advances in Cryptology - EuroCrypt ’87, Berlin, Springer-Verlag (1987) 117–126 Lecture Notes in Computer Science Volume 304.
6. Cohen Benaloh, J.: Secret sharing homomorphisms: keeping shares of a secret secret. In Advances in Cryptology - Crypto ’86, Berlin, Springer-Verlag (1986) 251–260 Lecture Notes in Computer Science Volume 263.

7. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Springer-Verlag (2001) 280–299
8. Burmester, M.V.D.: Homomorphisms of secret sharing schemes: a tool for verifiable signature sharing. In Advances in Cryptology - EuroCrypt '96, Berlin, Springer-Verlag (1996) 96–106 Lecture Notes in Computer Science Volume 1070.
9. Peng, K., Aditya, R., Boyd, C., Dawson, E., Lee, B.: Multiplicative homomorphic e-voting. In: Advances in Cryptology - Indocrypt '04, Berlin, Springer-Verlag (2004) 61–72 Lecture Notes in Computer Science Volume 3348.
10. Yokoo, M., Suzuki, K.: Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, ACM Press (2002) 112–119
11. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: Proceedings of 6th ACM Conference on Computer and Communications Security, ACM Press (1999) 120–127
12. Baudron, O., Stern, J.: Non-interactive private auctions. In: FC '01: Proceedings of the 5th International Conference on Financial Cryptography, Springer-Verlag (2002) 364–378
13. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45** (1998) 965–981
14. Hacigumus, H., Iyer, B.R., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database service provider model. In: SIGMOD Conference. (2002)
15. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Advances in Cryptology - Eurocrypt '04, Berlin, Springer-Verlag (2004) 1–19 Lecture Notes in Computer Science Volume 3027.
16. Sander, T., Tschudin, C.: Protecting mobile agent against malicious hosts. In: Mobile Agents and Security. LNCS 1419, Springer-Verlag (1998) 44–60
17. Gopalakrishnan, K., Memon, N., Vora, P.L.: Protocols for watermark verification. IEEE MultiMedia **8** (2001) 66–70
18. Schneier, B.: Applied Cryptography. John Wiley & Sons, Inc. (1996)
19. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. IEEE Trans. Softw. Eng. **22** (1996) 6–15
20. Anderson, R., Needham, R.: Robustness principles for public key protocols. In Advances in Cryptology - Crypto '95, Berlin, Springer-Verlag (1995) 236–247 Lecture Notes in Computer Science Volume 963.
21. Syverson, P.: Limitations on design principles for public key protocols. In: SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society (1996) 62
22. Chaum, D.: Blind signatures for untraceable payments. In Advances in Cryptology: Proceedings of Crypto '82, New York, USA, Plenum Publishing (1982) 199–204
23. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM **28** (1985) 1030–1044
24. Cox, I.J., Kilian, J., Leighton, T., Shamoon, T.: Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing **6** (1997) 1673–1687
25. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. In: Advances in Cryptology-CRYPTO'95. LNCS 963, Springer-Verlag (1995) 452–465

26. Blakley, G.R., Meadows, C., Purdy, G.B.: Fingerprinting long forgiving messages. In: Advances in Cryptology-CRYPTO'85. LNCS 218, Springer-Verlag (1986) 180–189
27. Pfitzmann, B., Schunter, M.: Asymmetric fingerprinting. In: Advances in Cryptology-EUROCRYPT'96. LNCS 1070, Springer-Verlag (1996) 85–95
28. Pfitzmann, B., Waidner, M.: Asymmetric fingerprinting for larger collusions. In: Proceedings of the 4th ACM Conference on Computer and Communicatios Security. (1997) 151–160
29. Domingo-Ferrer, J.: Anonymous fingerprinting of electronic information with automatic identification of redistributors. Electronics Letters **34** (1998) 1303–1304
30. Domingo-Ferrer, J.: Anonymous fingerprinting based on committed oblivious transfer. In Public key cryptography, PKC'99. LNCS 1560, Springer-Verlag (1999) 43–52
31. Pfitzmann, B., Sadeghi, A.: Coin-based anonymous fingerprinting. In: Advances in Cryptology - EUROCRYPT'99, Berlin, Springer-Verlag (1999) 150–164 Lecture Notes in Computer Science Volume 1592.
32. Sadeghi, A.R.: How to break a semi-anonymous fingerprinting scheme. In Information Hiding - IH'01, Berlin, Springer-Verlag (2001) 384–394 Lecture Notes in Computer Science Volume 2137.
33. Choi, J.G., Park, J.H., Kwon, K.R.: Analysis of cot-based fingerprinting schemes: New approach to design practical and secure fingerprinting scheme. In Information Hiding - IH'04, Berlin, Springer-Verlag (2004) 253–265 Lecture Notes in Computer Science Volume 3200.
34. Chaum, D., Damgaard, I.B., van de Graaf, J.: Multiparty computations ensuring privacy of each party's input and correctness of the result. In: Advances in Cryptology - CRYPTO'87. LNCS 293, Springer-Verlag (1988) 87–119
35. Crépeau, C., van de Graaf, J., Tapp, A.: Committed oblivious transfer and private multi-party computation. In: Advances in Cryptology-CRYPTO'95. LNCS 963, Springer-Verlag (1995) 110–123
36. Furon, T., Duhamel, P.: An asymmetric public detection watermarking technique. In: IH '99: Proceedings of the Third International Workshop on Information Hiding, Springer-Verlag (2000) 88–100
37. Fridrich, J.: Image encryption based on chaotic maps. In: Proceedings of the IEEE Conf. on Systems, Man, and Cybernetics. Volume 2., IEEE Press (1997) 1105–1110
38. Scharinger, J.: Fast encryption of image data using chaotic Kolmogorov flows. JEI **7** (1998) 318–325
39. Chuang, T., Lin, J.: New approach to image encryption. JEI **7** (1998) 350–356
40. Chang, C.C., Hwang, M.S., Chen, T.S.: A new encryption algorithm for image cryptosystems. J. Syst. Softw. **58** (2001) 83–91