An Asymmetric Traceability Scheme for Copyright Protection Without Trust Assumptions

Emmanouil Magkos[1], Panayiotis Kotzanikolaou[1], and Vassilios Chrissikopoulos[1]

[1] Department of Informatics, University of Pireaus, Greece
{emagos, chris}@unipi.gr
[2] Information Security Group, Royal Holloway, University of London, UK
mikeb@dcs.rhbnc.ac.uk

Abstract. Traceability schemes (also known as *traitor tracing* schemes) have been proposed as a method to establish copyright protection of broadcast information. With *asymmetric* traceability, the merchant cannot frame an innocent user, while no user can abuse the system without being detected. We propose an asymmetric solution for traceability, based on a very efficient symmetric scheme [4]. We do not make any trust assumptions about the broadcasting center or other authorities. Furthermore, we establish anonymity protection for all honest users: the identity of a user is protected, until a "fingerprint" of that user is found on a pirate decoder. We make use of well-known cryptographic techniques, such as *oblivious transfer*, *time-lock puzzles* and *blind signatures*

1. Introduction

With the rapid development of new IT technologies and electronic commerce, intellectual property for digital content has been an important issue. Many watermarking and fingerprinting techniques, mostly based on classical steganography, have been proposed [1]. While digital fingerprints help the content owner to identify a copyright violator (i.e., a "pirate"), digital watermarks give the means to prosecute the pirate. Especially for broadcast information, such as pay-per-view TV, web broadcast of online stock quotes, online databases and CD-ROM distribution, fingerprinting techniques have been combined with cryptography to enhance identification of redistributors. In these *traceability* schemes (also known as *traitor tracing* schemes [2]), the data supplier broadcasts encrypted information, and only the authorized users are able to decrypt it, by using their unique decryption keys. If some unauthorized users (pirates) get some decryption keys from a group of one or more authorized users (traitors) then the pirate decoder contains secret information that allows the data supplier to identify at least one traitor.

Copyright protection of digital data comes in three flavors. In the *symmetric case* (e.g., [2, 3, 5, 6]), the merchant knows the fingerprint (e.g. marking codes or decryption keys) that is uniquely linked with the buyer. If the merchant finds an illegally redistributed version of the original data, there are no means to prove to the Court that the buyer is guilty, since it could have been the merchant himself that tried to incriminate the buyer. In the *asymmetric case* (e.g., [13, 14, 15]), only the buyer knows the data with the fingerprint. If the merchant later finds a redistributed version of the data, he can identify the buyer and prove this to the Court. The *anonymous asymmetric case* (e.g., [16]) comes as a complementary protection for the buyer: Users buy information anonymously, but they can be identified if they redistribute the information illegally.

Recently, Kurosawa and Desmedt proposed two symmetric traceability schemes [4], where each buyer has only one decryption key. These schemes are very efficient and provably secure. For each of the two basic symmetric schemes, they also presented an asymmetric version. However, asymmetry is based on the existence of some trusted entities, i.e. an arbiter or trusted agents. These entities have knowledge of the decryption keys of all the users of the system, but they are trusted not to frame any user. We believe that the trust granted to these entities is unacceptable: there should be a protocol, which, if executed between the broadcasting center and a user, would establish asymmetry without the involvement of a third party. This protocol should also be secure against a misbehaving center or/and a malicious user.

Our Contribution. In this paper we turn the very efficient traceability scheme of Kurosawa-Desmedt [4] into an asymmetric scheme, without assuming the involvement of a trusted entity. For this reason we make use of a cryptographic technique called *oblivious transfer*. In addition, we propose a special *cut-and-choose* technique that assures, with a non-negligible probability, the correctness of the private keys that are obliviously transferred to the users of the system. Our solution can directly be embedded in the key generation procedure of the KD scheme [4]. Furthermore, our solution offers extra anonymity protection for the buyer, i.e., the KD scheme is turned into an anonymous asymmetric scheme. In order to establish anonymity, we make use of cryptographic tools and techniques that are publicly known and have been proposed during the past years, namely *time-lock puzzles* and *blind signatures*.

1.1 Related Bibliography

The first traceability schemes in the literature are due to Fiat and Naor [2]. However, these schemes are very inefficient: every user personal key consists of $O(k^2 \log n)$ decryption keys and the data supplier has to broadcast $O(k^4 \log n)$ ciphertexts. Recently, Kurosawa and Desmedt proposed very efficient traceability schemes [4], the KD schemes. They first derived lower bounds and proposed an optimum one-time scheme that satisfies these bounds. More recently, Boneh and Franklin presented a traceability scheme [6], the BF scheme, which is provably secure and achieves *full-traceability*, i.e., there is a tracing algorithm that catches all traitors that participate in the construction of a pirate decoder. In addition, their scheme offers *black-box traceability*, i.e., the pirate decoder is not opened but only queried in order to identify the traitors. They also describe a linear attack against the KD tracing algorithm [4], where two or more traitors are able to construct a pirate decoder that cannot be traced back to them. This attack was later addressed by Kurosawa, Desmedt and Burmester [7]. The new tracing algorithm presented in [7], makes the KD scheme full-traceable and black-box traceable. In addition, the gap between the ciphertexts and the plaintexts in the BF scheme is greater than the lower bounds achieved in the KD scheme. As a result, and to the best of our knowledge, the most efficient traceability scheme in the literature is the KD optimum traceability scheme.

The Asymmetric case. Pfitzmann [14] combined the Fiat-Naor symmetric scheme with a two party protocol [18] in order to turn it into an asymmetric scheme. Later, Pfitzmann and Waidner [15] got the same result by using another two party protocol [19] and secure cryptographic commitments [20]. However, their schemes are not efficient because the symmetric scheme of [2], on which they are based, is very inefficient.

This paper is organized as follows. Section 2 describes the basic building blocks required for our asymmetric scheme. In Section 3, an anonymous asymmetric version of the KD scheme is presented. Section 4 concludes the paper.

2. Building Blocks

2.1 The Kurosawa-Desmedt one-time traceability scheme

In [4], the data supplier T chooses a uniformly random polynomial $f(x) = a_0 + a_1 x + .. + a_k x^k$ over $GF(q)$ as the encryption key e_T, where q is a prime with $q > n$, for a set of n authorized users.

Key Generation. T gives to each authorized user u_i the personal decryption key $e_i = < i, f(i) >, i = 1,2,...,n.$

Encryption: *T* encrypts a session key *s*, as: $h = (h_0, h_1, .., h_k) = (s + a_0, a_1, .., a_k)$. Then, *T* broadcasts *h* to all users of the system.

Decryption: From the header *h* and the decryption key e_i, each user u_i decrypts *h* to compute *s* as: $(h_0 + h_1 i + .. + h_k i^k) - f(i) = s$.

Tracing: When a pirate decoder is confiscated, the pirate key e_p is exposed. If e_p contains $(j, f(j))$ for some user u_j, then *T* decides that user u_j is a traitor.

2.2 Non-interactive Oblivious Transfer

In [11], non-interactive oblivious transfer is described as follows: "*Bob has two strings S_0 and S_1. As a function of these and Alice's public key P_A, he computes a message OT(S_0, S_1) and sends it to Alice. Using her secret key x, Alice can extract from OT(S_0, S_1) exactly one of the strings S_0 and S_1, Bob will not know which of the two Alice got*". We will now summarize the protocol, as presented in [11, Section 2.1].

Setup. Let p be a prime and g be a generator of Z_p^*. $C \in Z_p^*$ is a globally known parameter of the system and is a number that it does not have discrete logarithm modulo *p*. Ways of finding such numbers are described in [11].

Key Generation. Alice chooses $x \in \{0,.., p-2\}$ at random and sets $a_0 = g^x$ and $a_1 = C(g^x)^{-1}$. Her public key is (a_0, a_1) and his secret key is *x*. Correctness of Alice's public key is achieved by verifying that $a_0 a_1 = C$.

Oblivious Transfer. Bob chooses $y_0, y_1 \in \{0,.., p-2\}$ at random and computes $\beta_0 = g^{y_0}$, $\beta_1 = g^{y_1}$. He also uses Alice's public keys to compute $\gamma_0 = \alpha_0^{y_0}$ and $\gamma_1 = \alpha_1^{y_1}$. Finally, he computes $r_0 = S_0 \oplus \gamma_0$ and $r_1 = S_1 \oplus \gamma_1$, and sends $OT(S_0, S_1) = (\beta_0, \beta_1, r_0, r_1)$ to Alice. On receiving β_0 and β_1, Alice uses her secret key to compute $\beta_i^x = \gamma_i$. Finally, she computes $\gamma_i \oplus r_i = S_i$. According to the Diffie-Hellman assumption [12], Alice will be able to produce γ_0 or γ_1, but never both. So, Alice is finally left with exactly one-out-of-two secrets and Bob does not know which secret Alice has ended with.

2.3 Time-lock Puzzles

With time-lock puzzles [8], Alice can encrypt a message *M* so that Bob can decrypt it after a period of *T* seconds, with *T* be a real (not CPU) time period. There are several ways to implement time-lock Puzzles. In [8], the message is encrypted with an appropriately large symmetric key, which in turn is encrypted in such a way that Bob can decrypt it only by performing *t=TS* squarings sequentially, where *S* is the number of squarings per second that Bob can perform. The computational problem of performing these squarings (each time squaring the previous result) is not parallelizable in any way: having two processors is no better than having one

2.4 Blind Signatures

Blind Signatures [9,10] are the equivalent of signing carbon-paper-lined envelopes. A user seals a slip of a paper inside such an envelope, which is later signed on the outside. When the envelope is opened, the slip will bear the carbon image of the signature. In order to establish correctness in a blind signature protocol, a cut-and-choose technique can be used: Alice sends *m* blinded messages to Bob, then un-blinds any *m-1* indicated by Bob. Bob signs the remaining message. There is a tradeoff between choosing a large *m* (strong correctness), and a small *m* (efficiency).

3. An anonymous asymmetric traitor tracing scheme

We present an asymmetric version of the KD scheme (see Fig. 1), by transforming its key generation procedure into a 2-round protocol between any user of the system, say Alice, and the Data Supplier. We do not make any trust assumptions about the Data Supplier or another entity. The basic cryptographic primitive that we make use of is an *oblivious transfer* of secrets (see Section 2.2): The Data Supplier associates Alice with two unique KD decryption keys, i.e., $S_0 =< i, f(i) >$, $S_1 =< j, f(j) >$. He then sends these secrets obliviously to Alice, in such a way that Alice finds out exactly one of the two KD decryption keys, S_0 or S_1. The Data Supplier does not know which key Alice ended with, and Alice does not know the value of the other key. If the Data Supplier wants to incriminate Alice, then he can construct a fake pirate decoder and randomly select one of the two keys that were obliviously transferred to Alice. The Data Supplier will incriminate Alice with a success probability of 1/2. We expect that the Data Supplier will not risk of making a false accusation, as Alice can always prove in front of a judge that her decoder contains a decryption key other than what the pirate decoder contains. In such case, the cost of the Data Supplier's false accusations Alice will be significantly greater than the gain from a successful framing.

Remark 1. Instead of using a (1-2) oblivious transfer protocol, a (1-*N*) protocol could be used [23], where Alice would choose one out of *N* messages. The choice of *N* implies a trade off between correctness and efficiency. In such a case, the probability of success for a false accusation on behalf of the data supplier would be equal to 1/*N*.

We also achieve anonymity for Alice: the Data Supplier does not know the identity of Alice, when she asks for a KD decryption key. Instead, he possesses a time-lock puzzle of Alice's identity, which can be used in case of Alice being a traitor. The use of the time-lock puzzle offers an extra protection for Alice. If the Data Supplier wants to incriminate Alice, then he must start solving very difficult computational problems, i.e., the time-lock puzzles of the users that apply for a decryption key. Even if the Data Supplier has the time and the computational power to break a puzzle that points to Alice, he still has at most 50% probability of incriminating her, because of the oblivious transfer mechanism (see also Remark 1).

Executing the protocol. In Figure 1, Alice creates a private/public key pair (x, a_0, a_1) for the oblivious transfer protocol and a time-lock puzzle of her real identity, T_A. Alice blinds both (a_0, a_1) and T_A, then signs the blinded messages with her signature key[1] and sends the result to the Data Supplier. The Data Supplier checks the correctness of the blinded messages (see also Section 2.2), signs them and returns them to Alice (Step 1).

In Step 2, Alice un-blinds M_0, M_1, M_2 to obtain $A_0 = (\mathrm{a}_0)^{SK_D}$, $A_1 = (\mathrm{a}_1)^{SK_D}$, $T_{AD} = (T_A)^{SK_D}$, where SK_D is the signature key of the Data Supplier. Then she sends A_0, A_1, T_{AD} and her public key (a_0, a_1) to the Data Supplier through an anonymous channel[2] (Step 3). The Data Supplier verifies his signature on a_0 and a_1 and uses them to prepare an instance for the oblivious transfer protocol, $OT(S_0, S_1)$, where S_0, S_1 are two KD decryption keys. In Step 4, Alice executes $OT(S_0, S_1)$ and finds out exactly one of the two KD keys, while the Data Supplier is not able to determine which key Alice extracted.

---

[1] There is a *Certificate Infrastructure* and users are legally bound by their signature. Mechanisms to establish non-repudiation for digitally signed messages are discussed in [22].

[2] There is an *anonymous* channel where users can send/accept messages that cannot be traced (e.g., by using traffic analysis). For example, e-mail anonymity can be established using *Mixmaster* re-mailers[2] [20]. HTTP anonymity can be established by using services such as the *Onion-Routing* system [21].

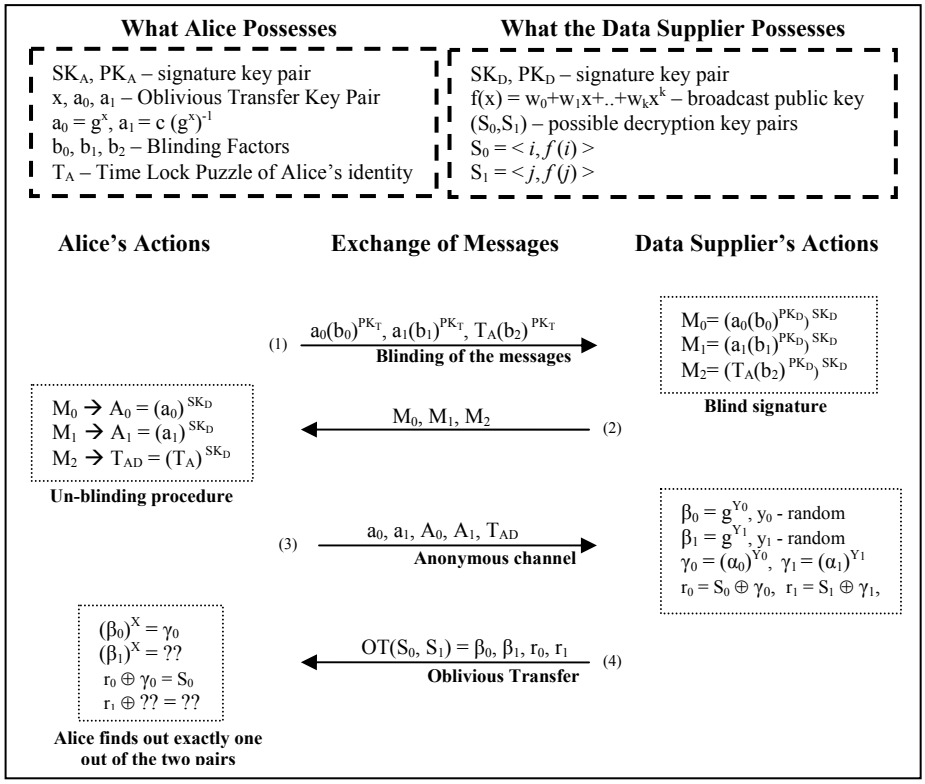

Fig. 1. An asymmetric key generation protocol for the KD scheme

Tracing. When a pirate decoder is confiscated, the private decryption key e_u of an unknown user u is exposed. The Data Supplier solves the time-lock puzzle[3] assigned with the decryption key, identifies the user u, decides that user u is a traitor and presents the credentials of the user's guilt in front of a judge. Note that if more than on traitors have been participated in the construction of the pirate decoder, then the new tracing algorithm of [7] for the KD scheme can be applied.

Remark 2. If Alice is a traitor and the Court accepts the credentials presented by the Data Supplier, then, apart from the penalties imposed by the laws for copyright violation, Alice will also be subject to a penalty price that is analog to the cost for the solution of the time-lock puzzle.

3.1 Assuring the Correctness of the KD Decryption Keys

In the protocol presented above, the Data Supplier is supposed to select correct decryption keys as input for the oblivious transfer protocol with Alice. The Data Supplier does not know which key Alice possesses, since he obliviously transferred two different KD decryption keys to Alice, and Alice selected exactly one of these keys. But, what happens if the Data Supplier uses two identical KD keys as input for the oblivious transfer protocol? Or, even worse, what happens if the Data Supplier obliviously transfers the same key(s) to another eligible user?

If the Data Supplier used two identical keys as input for the oblivious transfer protocol with Alice, then the Data Supplier would finally get to know Alice's decryption key, since

[3] The time T, needed for a puzzle to be solved, must be the same for all users. If necessary, cut-and-choose techniques can be used to assure that all puzzles have been constructed with the correct time information. These techniques are similar to those used in blind signatures.

Alice would choose one-out-of two identical keys. In such case, asymmetry would be broken and the Data Supplier could frame Alice.

On the other hand, if there is no way to assure that each instance of the oblivious transfer protocol contains two keys that are not re-used in any other instance for a different user, then there is a non-negligible probability that two legitimate users will extract the same KD decryption key, by executing different oblivious transfers. In case of dispute, it would not be clear if an innocent user is falsely accused as a traitor or if the user is actually a traitor. Thus, traitor tracing would not be possible.

For the rest of the section, we describe cut-and-choose techniques that force the Data Supplier to choose correct decryption keys for each oblivious transfer. Clearly, the following conditions are critical for the security of the system:

Condition A. For each user *i*, *i*=1..n, the instance of the oblivious transfer protocol $OT_i(S_{i,0}, S_{i,1})$ must contain two different KD decryption keys, i.e., $S_{i,0} \neq S_{i,1}$.

Condition B. For two different users *i*, *j*, the instance $OT_j(S_{j,0}, S_{j,1})$ of the oblivious transfer protocol, must contain different KD decryption keys from the instance $OT_i(S_{i,0}, S_{i,1})$, i.e., $S_{i,0} \neq S_{i,1} \neq S_{j,0} \neq S_{j,1}$.

3.1.1 Forcing the Data Supplier to use different KD keys in each oblivious transfer

We propose a cut-and-choose technique, which can be applied on the oblivious transfer protocol to ensure that for each user, the Data Supplier obliviously transfers two different KD keys, with overwhelming probability. We denote the encryption of a message *m* with a key *X* as $Encr_X(m)$.

Let $OT_i =< Encr_{K_i}(S_{i,0}), Encr_{K_i}(S_{i,1}) >,\ i \in (0,1)$, be two instances of the oblivious transfer protocol (see step 4 - Fig.1), where $S_{i,0}$, $S_{i,1}$ are four different KD decryption keys and K_i are two encryption keys of a deterministic symmetric encryption scheme (*e.g.* DES keys). Moreover, let C_{K_i} be a commitment on the key K_i. The commitment could be the output of a hash function (*e.g.* MD5) applied over each key K_i. The Data Supplier *T* sends (OT_0, C_{K_0} || OT_1, C_{K_1}) to Alice, along with a signature of that message.

Phase 1: Checking the correctness of the encryptions. Alice randomly selects one of the two instances (say without loss of generality OT_0) and *T* reveals all secret information for this instance (i.e., $y_{0,0}, y_{0,1}, \gamma_{0,0}, \gamma_{0,1}$) in order to prove that it is well constructed. Given this information, Alice retrieves both $Encr_{K_0}(S_{0,0})$ and $Encr_{K_1}(S_{1,1})$. She compares these values and if $Encr_{K_0}(S_{0,0}) \neq Encr_{K_1}(S_{1,1})$ Alice is convinced that $S_{0,0} \neq S_{0,1}$ since the same key has been used to encrypt both KD keys and the symmetric encryption scheme is deterministic. If the encrypted values are equal, then this means that the cleartext values $S_{0,0}$ and $S_{0,1}$ are equal too, so *T* has cheated and is reported to the judge.

Phase 2: Checking the correctness of the symmetric key. If the checks in phase 1 are correct, Alice executes the remaining oblivious transfer. In our example, this would be OT_1. Thus, Alice will extract either $Encr_{K_1}(S_{1,0})$ or $Encr_{K_1}(S_{1,1})$. At this time, *T* is asked to send the corresponding symmetric key K_1 to Alice. Alice checks the correctness of K_1 in two steps:

first, she re-constructs the one-way commitment C_{K_1} to verify that she has been given the key the data supplier had committed to. Second, she uses it to decrypt the encrypted outcome, and see if a valid KD key is generated. If either check fails, *T* has cheated and is reported to the judge; Otherwise, *T* has followed the protocol with overwhelming probability and Alice has used K_1 to obtain an official KD key.

Observe that a misbehaving Data Supplier has a non-negligible probability of getting caught, since he does not know *a priori* which instance he will be asked to open in phase 1, and which instance Alice will execute in phase 2. Thus, if one of the two instances in not well constructed, then *T*'s misbehavior will be revealed with probability ½ for each user.

Note that *T* could try to cheat by encrypting two identical KD keys $S=S^*$ with two different symmetric keys $K \neq K^*$. In such case, $Encr_K(S) \neq Encr_{K^*}(S^*)$ while $S=S^*$, and the check in phase 1 will succeed. However, *T* has already committed to either K or K^* and cannot *a priori* determine which of the two encryptions Alice will extract from the oblivious transfer protocol. Thus, *T*'s misbehavior will be revealed with probability ½ for each user.

Remark 3. Because the keys are encrypted, there is no wasting of the pair of keys $S_{0,0}$ and $S_{0,1}$ that where used in the instance OT_0 that was revealed to Alice in phase 1. Note that *T* does not send to Alice the symmetric encryption key K_1 because in that case Alice would decrypt $Encr_{K_0}(S_{0,0})$ and $Encr_{K_0}(S_{0,1})$ to obtain both keys. So, the Data Supplier can later use this pair of KD keys for another user, since none of these keys has been revealed to Alice. However, *T* should re-encrypt the KD keys with a different symmetric key.

3.1.2 Forcing the Data Supplier to use different KD keys for different oblivious transfers

In order to force the Data Supplier to use different KD keys for different oblivious transfers, we make use of a bulletin board. Our technique can be seen as an extension of the technique used in Section 3.1. We require that each instance of the oblivious transfer also contain the hash values of the KD keys, i.e.:

$$OT_i = (C_{K_i}, Encr_{K_i}(S_{i,0}), Encr_{K_i}(S_{i,1}), hash(S_{i,0}), hash(S_{i,1}))$$

After the protocol of Section 3.1 has been executed, T publishes both hash values $hash(S_{i,0}), hash(S_{i,1})$ to the bulletin board. T also publishes a statement that the KD keys that correspond to these hash values have been registered to an authorized user. Alice, who has been left with exactly one KD key, hashes that key and checks if the result matches with the corresponding hash value, contained in OT_i. If not, *T* is reported to the Judge. Alice also checks the board to see if the hash value of her KD key has been registered. If the hash of the KD key appears to have been registered more than one times, *T* is reported for cheating.

The possibility that *T* has included fake hash values of the KD keys in each OT_i, and gets away with it is less than ½. The analysis is the same as in Section 3.1.

References

[1] Peticolas, F., Anderson, R., Kuhn, M., "Information Hiding- A Survey", Proceedings of the IEEE, special issue on protection of multimedia content, IEEE Vol. 87(7), 1999, pp. 1062-1078.

[2] Chor, B., Fiat, A., Naor, M., "Tracing Trators", Advances in cryptology – CRYPTO 94, LNCS 293, Springer-Verlag, 1994, pp. 257-270.

[3] Boneh, D., Shaw, J., "Collusion Secure Fingerprinting For Digital Data", Advances in cryptology – CRYPTO 95, LNCS 963, Springer-Verlag, 1995, pp. 452-465.

[4] Kurosawa, K., Demedt, Y., "Optimum traitor tracing", Advances in cryptology – EUROCRYPT 98, LNCS 1403, Springer-Verlag, 1999, pp. 145-157..

[5] Stinson, D., Wei, R., "Combinatorial properties and constructions for traceability schemes", SIAM Journal on Discrete Mathematics, Vol. 11(1), 1998, pp. 41-53.

[6] Boneh, D., Franklin, M., "An Efficient public Key Traitor Tracing Scheme" Advances in cryptology – EUROCRYPT 90, LNCS 1666, Springer-Verlag, 1999, pp. 338-353.

[7] Kurosawa, K., Burmester, M., Demedt, Y., "The failure of the Boneh-Franklin / Stinson-Wei attack against optimal traitor tracing", DIMACS 2000, 2000.

[8] Rivest, R., Shamir, A., Wagner, D., "Time-Lock Puzzles and Timed-Released Crypto", LCS Technical Memo MIT/LCS/TR-684, 1996, http://www.theory.lcs.mit.edu/~rivest/RivestShamirWagner-timelock.ps

[9] Chaum, D., "Blind Signatures for Untraceable Payments", Advances in Cryptology – CRYPTO 82, Plenum Press, pp. 199-203, 1982.

[10] Schneier, B., "Applied Cryptography – Protocols, Algorithms and Source Code in C", 2nd Edition, 1996.

[11] Bellare, M., Micali, S., "Non-Interactive Oblivious Transfer and Applications", Advances in cryptology – CRYPTO 89, LNCS, Springer-Verlag, 1990, pp. 544-557.

[12] W. Diffie, M. Hellman, "*New Directions in Cryptography*", IEEE Transactions on Information Theory IT-22, pp. 644-654, November 1976.

[13] Pfitzmann, B., Schunter, M., "Asymmetric Fingerprinting", Advances in cryptology – EUROCRYPT 96, LNCS 1070, Springer-Verlag, 1996, pp. 84-95.

[14] Pfitzmann, B., "Trials of Traced traitors", Information Hiding Workshop, LNCS 1174, Springer-Verlag, 1996, pp. 49-64.

[15] Pfitzmann, B., Waidner, M., "Asymmetric Fingerprinting for Larger Collusions", ACM Conference on Computer and Communication Security, ACM, 1997, pp. 151-160.

[16] Pfitzmann, B., Waidner, M., "Anonymous Fingerprinting", Advances in cryptology – EUROCRYPT 97, LNCS 1233, Springer-Verlag, 1997, pp. 88-102.

[18] Chaum, D., Damgard, I., Graaf, J., "Multiparty Computations ensuring privacy of each party's input and correctness of the result", Advances in cryptology – CRYPTO 87, LNCS 293, Springer-Verlag, 1988, pp. 87-119.

[19] Goldreich, O., Micali, S., Wigderson, A., "How to play any mental game – or – a completeness theorem for protocols with honest majority", 19th STOC, 1987, pp.218-229.

[20] Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, Vol. 24(2), (1981) 84-88

[21] Goldschlag, D., Reed, M., Syverson, P.: Onion Routing for Anonymous and Private Communications. Communications of the ACM, Vol. 42(2), (1999) 39-41

[22] You, C., Zhou, J., Lam, K.: On the Efficient Implementation of Fair Non-Repudiation. In: Proceedings of the 1997 IEEE Computer Security Foundations Workshop. IEEE CS Press (1997) 126-132

[23] Naor, M., Pinkas, B., "Oblivious Transfer and Polynomial Evaluation". In: Proceedings of the 31th ACM Symposium on Theory of Computing, ACM (1999), pp. 245-254.