Copyright Protection with Traitor Tracing schemes and supplying undeniable proof of the traitor's treachery by means of Asymmetric Traitor Tracing

Agnieszka Dembczynska and Pawel Kaniewski

Departament of Computer Science,
Linköping Institute of Technology, Sweden
agnde942@student.liu.se
pawka317@student.liu.se

Abstract

This work should show how traitor tracing can achieve the goal of piracy prevention and which measures can be taken against traitors and pirates in consideration of different assumptions. In the first part we define traitor tracing, its goals, usage and overview some of cryptographic schemes that help to trace the source of the leak. In the second part we go more detailed into asymmetric public-key traitor tracing, one of the techniques that support non-repudiation. Implementation of such scheme provides undeniable proof of the implication of the traitor subscribers. Therefore it gives the provider - in the context of pay television – more possibilities to take measures against the traitor/pirate.

1. Introduction

Traitor tracing schemes provide protection against illegal access to digital data of diverse types, including computer software and multimedia objects**.** The secure distribution of digital content stream to an exclusive set of subscribers is an important aim that should be achieved in the entertainment industry. A very relevant application is in the context of Pay-TV. A stream of content, which is broadcast through a network, is addressed to some privileged users. Only they should be able to view certain programs. In this application, the programs are normally encrypted. To get the content the subscriber need a decoder and some sensitive data – decryption keys. However it is desirable by catching any pirate decoder to reveal the source of its decryption keys. A combination of a traitor tracing scheme and a broadcast encryption scheme is a very powerful tool. When a traitor is traced, the subset of legitimate users can be changed by simply dropping the traced traitor from it.

There are two different components in fighting piracy. Firstly identifying that piracy is going on and preventing the further transmissions of information to the pirate (excluding of recipient). Secondly identifying the source of the piracy and taking the measure against it. Here go traitor tracing schemes into action, they help in three aspects of piracy prevention: they deter users from cooperating with pirating, they identify the pirates and enable to take the legal actions against them. So they can be used to disable active pirates users. However the symmetric traitor tracing schemes do not support the non-repudiation and therefore cannot supply the undeniable evidence of the traitors' treachery. The accused traitor can always defend himself by claiming, that the dishonest merchant has framed him by reusing his personal decryption key. Actually this really can happen, as long as the merchant knows the decryption key of the users. This is the reason why the asymmetric traitor tracing schemes, that support non-repudiation and supply the undeniable proof of the traitors fault, could be very useful. We will describe those schemes more detailed in the second part of our assignment.

1.1 Some definitions

Traitor tracing schemes have the desirable property that identifying traitor can be achieved by considering the pirate decryption process as a black box. In order to identify a traitor, it suffices to capture one pirate decoder and examine its behavior. We use the term "pirate decoder" to represent the pirate decryption process.

The traitor we call an authorized user of the system who allows the third party to obtain the data by for example leaking (some of) their key information. A pirate is the non-authorized party that uses subscriber-key information for illegal data reception.

Throughout the paper we name any content that should be protected from pirates by encryption as "cleartext" and already encrypted form as "ciphertext".

In today it is often considered sufficient to prevent piracy by supplying the legitimate users with special designed hardware that prevent interference and access to enclosed cryptographic keys (token, smartcards). We call such solution “secure hardware” or more precisely a “tamper-resistant box”.

1.2 Problems and challenges

Before any data supplier can take any measure against piracy it should first be desirable to determine the question of guilt. If only one person knows some secret the guilty party is evident. A more complex situation arises if a group of people has access to the secret and it next becomes public knowledge. Any data that should be kept secret from a unauthorized parties, while making it available to a certain set of users (e.g. paying subscribers of digital TV), can be protected by encryption. The subscribers are given keys that are required to decrypt the content. However it does not solve the problem above. If a pirate user has obtained content in encrypted form and all the keys that are required to decrypt it, there is technically not possible to prevent her from continuing to use the content. Thus the main aim should be to prevent traitors from distributing the keys that enable the decryption of encrypted content.

1.3 Sample practical applications

To prevent traitors form distributing their cryptographic keys the data supplier has different - more or less efficient – possibilities. We assume there are some television providers that encrypt the program before broadcasting them to the users. Every authorized user is given a different key for decrypting the ciphertext. Should the key used in a pirate decoder be discovered, it will be linked to a personal key of a traitor and this traitor will be identified. Obviously, a possible solution is to encrypt data separately under different personal keys. However it becomes impracticable in any broadcast environment for the very reason that the length of the ciphertext would grow n times the length of the cleartext, where n depicts the number of authorized users (as well as number of number of stored keys. It means inefficient managing of storage space by provider.). It is also very problematic in the context of online database as well as media storage like CD-ROM and DVD because this requires producing every copy different form each other.

The goal of a traitor tracing scheme is quite similar to what is often called fingerprinting with other types of data, except that it is not the broadcast data that is fingerprinted, but, at least in principle, a cryptographic key that is used to decrypt the broadcast data. It means simply that the cleartext redistribution is not addressed here. The reason for this restriction is that operating a pirate TV-broadcast station is too expensive and too risky. The formalization of it is that the traitors only redistribute information obtained during user initialization. This is a much stronger restriction; it permits that the traitors cannot be traced if they redistribute a tiny amount of data obtained after user initialization. Such a definition may fit applications where timelines are crucial and the traitors may not be able to redistribute even a tiny amount of data fast enough. Practical examples could be: services with current stock-exchange information, exchange rates of the currency, live transmissions from football games or others live sport presentations.

In our further consideration we rule simply out cases of piracy, which deal with redistribution of cleartext because it seems to be not effective. In all such cases transmitting the cleartext from the traitor to a pirate-user is rather expensive compared to the mass distribution channels the legal data supplier uses. It might also be – as mentioned above - the case, as with on-line databases or newspapers, that the data is continuously changing and therefore it is very hard for the pirate to keep an updated copy of data.

1.4 Related areas

1.4.1Broadcast encryption

The area of broadcast encryption has been studied by Fiat and Naor and has received much attention since then. It is intended for applications where an information provider broadcasts a lot of information in encrypted form, and only legitimate users are supposed to be able to decrypt it. The typically example is Pay-TV, if the information is actually transmitted over a broadcast channel. The broadcast encryption schemes in [1] have three phases:

- Provider initialization, where the information provider generates some information that he will need with all users. (initial records)
- User initialization, where an individual user registers with this information provider. The information that the user stores after this phase is called this user’s personal key.
- Session sending. The data are transmitted, divided into smaller parts called sessions. Each session is encrypted with a different session key, and some additional information is broadcast that allows all the legitimate users and nobody else to decrypt the session key with their personal keys, and thus to decrypt the real data.

Very important is that the model of broadcast encryption does not allow interaction between users, neither in initialization nor later. This is realistic for applications like Pay-TV, and it is one of the features that distinguish broadcast encryption from conference key distribution. It means that every end user has own set of keys, which he need to decrypt the data. Thus it should be guaranteed, to find this user, after his set of keys will be traced in any pirate decoder.

1.4.2 Fingerprinting
Fingerprinting is a technique extensively used by law enforcement to identify criminals when traces are left behind which uniquely match a suspect. Fingerprinting schemes are cryptologist mechanisms for the copyright protection of digital data. Buyers who redistribute the data illegitimately are called traitors. Fingerprinting schemes discourage traitors by enabling the original merchant of the data to identify the traitor who is the original owner of this copy.

Boneh and Shaw [4] have suggested a scheme for fingerprinting different copies of an electronic document by inserting a different watermark into each copy. The scheme has the desirable property that it does not allow generation of a new copy whose fingerprint does not reveal at least one of the copies that were used. It is possible to use the scheme in traitor tracing, but the number of keys that each user should have is quite large and there are other more efficient solutions.

2. Traitor Tracing Schemes

The notion of tracing system was first introduced by Chor, Fiat and Naor in [1], and was later refined to the Threshold Traitor model. Its goal is to distribute decryption keys to the users so as to allow the detection of the key that is used in any pirate decoder or on a new created key by colluding traitors using keys of at most "t" users. Black-box tracing assumes that only the outcome of the decoding box can be examined. Chor and Naor provide combinatorial and probabilistic constructions that guarantee tracing with high probability. The public key tracing scheme of Boneh and Franklin [2] provides a number-theoretic deterministic method for tracing.

2.1 Assumptions

2.1.1 Cleartext redistribution is not addressed.
In our further consideration we rule out cases of piracy, which deal with redistribution of cleartext because it seems to be not effective. It means simply that the cleartext redistribution is here not addressed.

2.1.2 No tamper-resistant box.
We do not assume that the personal keys are in tamper-resistant boxes, which would prevent the traitors from redistribution their secret information. The reason for that is that there are several methods that use hardware faults in order to reveal the keys that are enclosed inside. Furthermore the secure software solutions with good tamper-resistant boxes are deemed too expensive. Thus we disregard the possibility of application of any tamper-resist box and assume in our assignment that the traitors have found the access to their own personal keys and can redistribute them. The goal is to trace them if they do so, both for punishment after the fact and for deterring them from redistribution in the first place.

2.1.3 Underlying encryption scheme is secure
The common assumption is that it is hard to break the underlying encryption scheme so the pirates will rather try to obtain the key information to decrypt the scrambled message.

2.2 Different approaches to traitor tracing schemes

Fully resilient versus **threshold tracing schemes:** Traitors may conspire and give an unauthorized user (or users) a subset of their keys, so that the unauthorized user will also able to compute the real message key from the values he has been able to decrypt. The goal of the system designer is to assign keys to the users so that when a pirate decoder is captured it should be possible to detect at least one traitor. We distinguish between two kinds of tracing schemes. **Fully resilient schemes**, which can be used against any decoder which decrypts with non-negligible success probability (tracing of source of any pirate decoder), however in many application such security is not needed and it is enough to fight the pirate decoders which have a considerable success probability. (For example, in the pay-TV applications pirate decoders, which decrypt only a part of the content are probably useless). Thus there are **threshold schemes**, which trace decoders, which decrypt with probability greater than some threshold. (q, which is a parameter of the scheme. These schemes are considerably more efficient than fully resilient schemes.)

Within fully resilient and threshold tracing we can distinguish between one or two level tracing and between open and secret scheme. Open scheme treats circumstances where the decryption schemes used by all users are in the public domain, and the decryption keys themselves are the only information that is kept secret.

The secret type is where the actual decryption scheme as well as the keys are kept secret.

We can combine the different approaches to get different schemes, which can be used, as it is desirable. Thus we can obtain: fully resilient, open, one-level; fully resilient, open, two-level; fully resilient, secret, one-level; fully resilient, secret, two-level; threshold, one-level; threshold two-level. All the schemes are based on the usage of hash functions combined with any symetric cryptosystem and do not require of public key operations. The basic usage of hash functions is to assign decryption keys to authorized users. The two level schemes are more complicated but reduce the size of the enabling block. The fully resilient scheme has a short key length but the data redundancy overhead is quite large. The threshold schemes feature a tradeoff between the length of the personal key and the data redundancy overhead. It is possible to make one parameter very small by increasing the other parameter, and it is possible to achieve reasonable results for both measures.

Which of the previous schemes will be applied depend on the aim of the security scope and on the requirements (generally equipment) for an authorized user or requirements for the data supplier. The efficiency of the solution to fighting piracy can be measured in terms of several performance parameters. The parameters are the memory and computation requirements for both: an authorized user and the data supplier. It seems to get special importance if the user has limited computation and storage capabilities. Nowadays these parameters are less important for the data suppliers since they can use large storage space and perform its computation offline. A further efficiency parameter is the data redundancy overhead. It means the increase in data size that is needed in order to enable the tracing. This refers to the communication overhead (in broadcast or online systems) or the additional "wasted" storage on CD-ROM or DVD type systems.

Table 1 shows an example of complexity of different tracing traitors schemes.

Public Key Traitor Tracing: Boneh and Franklin [2] investigated public key traitor tracing schemes, which enable public key encryption and, being based on a number-theoretic assumption, are more efficient than combinatorial tracing (described below). In public-key traitor tracing, there is one authority that is responsible for the broadcasting infrastructure (which we call the system manager) and several, non-trusted, content-distributors that may take advantage of the public-key encryption procedure (published by the system manager) to distribute content to the subscribers of the system.

The combinatorial Tracing Schemes: Their properties were presented by Stinson and Wei in [10], and Staddon [11] investigated the relations between combinatorial tracing schemes and broadcast encryption schemes.

Self-enforcement schemes: In conjunction with identifying the traitor by copyright protection there has been suggested a lot of different schemes. Dwork, Lotspiech, and Naor [3] presented self-enforcement schemes, where the content is encrypted and each legitimate user receives a different decryption key which includes some sensitive information related to the user (e.g. his credit card number). Users will be not willing to disclose their keys to other since the keys contain this sensitive information. However this scheme is less efficient because of the size of the personal key and of the data redundancy.

Dynamic traitor tracing schemes: Finally, Fiat and Tassa [5] introduced dynamic tracing schemes in which, in order to locate the traitor, the tracing algorithm dynamically changes the content that is being broadcast to different subsets of the users. These schemes enable tracing even if the traitor is revealing the content itself and not only the keys that encrypt it.

2.3 Construction of traitor tracing schemes

The message in a traitor tracing scheme consists of an enabling block and cipher block. The cipher block is a part of the encrypted content of data (a few seconds of music or video clip) under some secret random key 's'. The enabling block allows authorized users to obtain key 's'. It consists of several encrypted elements and every user will be able to compute 's' by decrypting the values from the enabling block for which he has keys and then computing the actual key from these elements.

An adversary who wants to decrypt the message can either break the encryption scheme that was used in the cipher block (without using any information from the enabling block), or try to learn some information from the enabling block, that might help in the decryption process. The common assumption is that it is hard to break the underlying encryption scheme so one is interested in preventing attacks of the latter kind.

Table1. Examples of the complexity of different Tracing Traitors Schemes, using n=10^6, k=500, p=10-3, q=3/4,

p – Error probability, (probability that pirates cannot be traced.
q – Threshold parameter
n – Number of users
k – Number of traitors

	PROPERTY	PERSONAL KEY	DATA REDUNDANCY	DECRYPTION OPERATIONS
Trivial		1	1.000.000	1
Secret two-level	Best fully-res	493	11.300.000	493
Threshold	One-level, min. data redun.	26.500	2000	1
Threshold	Two-level, min.key	1.570	82.000	8
Threshold	Two-level min.key	370	574.000	12
Threshold	tradeoff	6.300	27.500	3

We restrict ourselves to the so-called one-level schemes from the Chor-Fiat-Naor Schemes. They consist of the following common elements:

- Provider initialization, where the information provider generates *l* sets of *b* keys each. These keys also belong to the given symmetric encryption scheme. We call each such set a *bucket*. The keys in the *i*-th bucket are denoted: $key_{i,1}$, $key_{i,2}$, …$key_{l,b}$.
- User initialization, where each user gets one key from each bucket. We call the indices of these keys the user's *codeword*.
- Session key s is the xor (the bitwise exclusive or), of l random values s_1, s_2, …s_l, which we call *shares* of the session key.
- The enabling block contains for all i, the encryptions of s_i under each key of the i-th bucket. Each user that has one key from the *i*-th bucket, can decrypt one of these b encryptions of the *i*-th *share* and finally reconstruct the session key *s*.

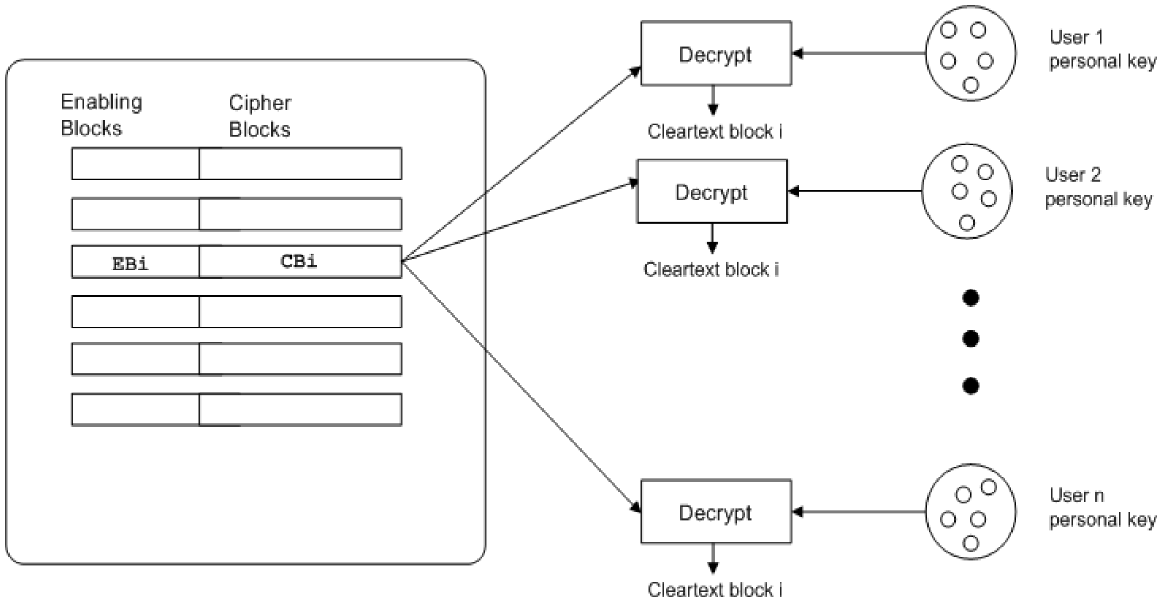

Fig.1. A high-level view of the traitor tracing scheme

l \ b	1	2	…	b	
1	key $_{1,1}$	key $_{1,2}$	…	key $_{1,b}$	set / *bucket*
2	key $_{2,1}$	key $_{2,2}$	…	key $_{2,b}$	
…	…	…	…	…	
l	key $_{l,1}$	key $_{l,2}$	…	key $_{l,b}$	

Fig.2. Keys for the simple one-level scheme.

3. Asymmetric traitor tracing

The traitor tracing schemes as we described in the first part of our assignment are intended for tracing people who betray the content provider by giving away their decryption keys to the unauthorized users (pirates). These schemes should provide an evidence of the traitor's treachery, thus allowing the content provider to take some actions against the dishonest user. However, the symmetric traitor tracing schemes cannot provide the undeniable proof of the traitor's fault that could unambiguously convince a third party. The reason for this is the symmetry assumption underlying the scheme. The legitimate users share all their secrets with the information provider so it is not possible to distinguish the source of the leak. The seemingly redistributed information could just as well be produced by the dishonest user, as by the content provider himself (or some of his employees). Actually this property seems very undesirable in the context of the efficiency of the tracing scenario against piracy. It prevents the system manager from pressing criminal charges against subscribers that leak their key-information, thus significantly lowering the commercial viability of piracy. In our opinion this is a very important aspect of the piracy problem in conjunction with digital content distribution. In the times, when key information can be easily redistributed on the Internet with only low probability of being traced, the seriousness of possible consequences should be the factor that deters users from treachery. For this reason we want to focus in our assignment on asymmetric traitor tracing algorithms and how they can be used against the illegal users of the digital content distribution systems.

3.1 Assumptions

In the asymmetric variant of traitor tracing the merchant is not considered as trusted and may attempt to frame a buyer by embedding the buyer's codeword in a second copy of the object. Thus the tracing procedure must produce undeniable proof of the implication of the traitor subscribers. In this scheme the content provider that finds the pirate decoder is confronted with information that he could not have produced on his own.

3.1.1 The goal: non-repudiation

The underlying assumption of asymmetry should provide for achieving non-repudiation. The tracing algorithm should produce a solid proof for the implication of the traitor that could convince a third party of the user's guilt. Such a proof lets the content provider not only take some unilateral measures against the traitor (e.g. disconnect him from the service) but is also a base for pressing criminal charges.

3.1.2 First asymmetric traitor tracing schemes

The existence of asymmetric traitor tracing scheme was first shown by Pfitzmann in [6], who also introduced the setting of asymmetric traitor tracing. She presented several example constructions of asymmetric traitor tracing based on combining the symmetric scheme with a protocol for secure two party computation. However these schemes were of little practical importance because of their inefficiency for practical use.

Later another scheme was presented by Kurosawa and Desmedt in [7]. They make use of threshold mechanism to ensure the non-repudiation property. In this scheme the capability to implicate the innocent user is shared between a number of authorities.

In [12] the involvement of trusted third parties has been eliminated. This scheme uses the oblivious polynomial evaluation and it was meant to be the first concrete construction of a practical asymmetric public-key traitor tracing scheme that does not rely on trusted agents. However, a flaw has been shown in this scheme by Kiayias and Yung in [8], who have provided a proof that "Any collusion of traitors of more than a single user can generate keys that are not traceable in the scheme of [12]".

In the same paper the authors also break another asymmetric scheme, presented in [9]. They claim in some circumstances tracing in the scheme of [9] would require exponential time, "as the tracer will have to use in the decoding algorithm all possible values of the underlying finite field Z_q which is exponentially large (the size of an element in the underlying finite field coincides with the security parameter of the system).”

Finally most our attention has been focused on the already mentioned [8]. Authors of the scheme not only critically analyze the already existing schemes, break their efficiency claims (by providing convincing proofs), but also present their own suggestion for efficient asymmetric public-key traitor tracing scheme, that should be comparable in efficiency to previous non-asymmetric schemes. The efficiency has been proved in the “non-black box” traitor tracing model. We want to present the idea of this proposition as an example of an asymmetric traitor tracing scheme, because we believe that this is the most efficient and practical solution suggested so far in this area.

3.2 Preliminaries

3.2.1 Decisional Diffie Hellman

The scheme we present assumes working in a multiplicative cyclic group G[1] of large prime order[2] over which solving the Decisional Diffie Hellman (DDH) problem is hard.

The hardness of computing discrete logarithms in some large finite groups has been the basis for many cryptographic schemes and protocols, starting form the Diffie-Hellman key exchange protocol, and continuing with encryption and signatures schemes, as well as protocols for numerous other applications. However, since the unconditional statements regarding the computational hardness of computing discrete logarithms could not been proved, mathematical assumptions regarding the computational hardness of this set of problems are formulated. Based on these assumptions properties of the protocols are proved. One of those assumptions is Decisional Diffie Hellman, formulated as follows:

The Decisional Diffie-Hellman (DDH) Problem: Given a group G, a generator[3] g of G, and three elements a, b, c $\in$ G, decide whether there exist integers x, y such that $a = g^x$, $b = g^y$, and $c = g^{xy}$.

The Decisional Diffie-Hellman (DDH) Assumption: Any probabilistic polynomial time algorithm solves the DDH problem only with negligible probability.

For example G can be the subgroup of order q of Z_p*, where q | p – 1 and p, q are large primes. In the following g will denote a generator of G. Arithmetic in the exponents is performed in the finite field Z_q.
Let $h_0, h_1, \dots, h_v$ be random elements of G so that $h_j := g^{r_j}$ for j = 0, … , v. For a certain element $y := g^b$ of G a representation of y with respect to the base $h_0, \dots, h_v$ is a (v + 1) - vector $\delta := < \delta_0, \dots, \delta_v >$ such that $y = h_0^{\delta_0} \dots h_v^{\delta_v}$, or equivalently $\delta \cdot r = b$ where $\cdot$ denotes the inner product between two vectors. Obtaining representations of a given y w.r.t. some given base $h_0, \dots, h_v$ is as hard as the discrete-log problem over G.

3.2.2 Oblivious Polynomial Evaluation

The basic building block to achieve asymmetry and non-repudiation property in [8] is oblivious polynomial evaluation (OPE).

An OPE protocol involves two parties:

- The sender S, who possesses a secret polynomial $P \in Z_q[x]$,
- The receiver R, who possesses a secret value $\alpha \in Z_q$.

The protocol allows the receiver to compute the evaluation of the sender’s polynomial P over its secret value α in such a way that:

- The sender S cannot extract any non-trivial information about the value α.
- The receiver R cannot extract any information about the polynomial P, other than what can be trivially extracted from the value P(α).

In [8] a two communication flow protocol is assumed, where {OPE}(α) denotes the data transmitted by the receiver R to the sender S in the first flow, and {OPE}(P(α)) denotes the data transmitted by the sender to the receiver in the second communication flow. According to the properties of oblivious polynomial evaluation {OPE}(α) does not yield any non-trivial information about the value α, and {OPE}(P(α)) contains enough information for the receiver to compute P(α) but not any further non-trivial information about the secret polynomial P.

Additionally the OPE used in [8] has two properties:

1. It is malleable: given {OPE}(α) the sender can easily compute {OPE}(α + α’), for a random α’.

[1] **Def**. The multiplicative group Z_m is the set of elements in Z_m that are relatively prime to m. It is denoted by Z_m*. If p is prime then Z_p* = {1, … , p – 1}. For every prime p, Z_p* is cyclic.
[2] **Def.** For a $\in Z_m$*, the order of a is the smallest positive integer t such that $a^t = 1 \pmod m$.
[3] If there exists an element α $\in Z_n$* whose order is σ(n) then this element is called a primitive element, or a generator of Z_n*, and Z_n* is said to be cyclic.

2. It is performed over a publicly committed value (α can be thought of as a private key whose public key is publicly known).

3.3 General scheme of asymmetric public-key traitor tracing

Asymmetric Traitor Tracing scheme usually involves following parties:

- the system-manager – the system-manager is responsible for administrating the system, issuing subscriber information and tracing pirate devices.
- the subscribers (users)
- the content providers – merchants that use the system to distribute encrypted data to some set of subscribers
- the judge – a third party (not necessarily trusted) that will perform the verification of the proof provided by the system-manager. On the basis of the verification results the judge should decide of the user's guilt or innocence.

Asymmetric traitor tracing scheme consists of following elements:

- Join.
  Initialization of the new user. As the result of the protocol each new user will be assigned a personal key. The personal key should be assigned in such a way that it guarantees non-repudiation on the one hand, but is not entirely known to the system-manager on the other (otherwise the asymmetry would be broken).
- Encryption.
  The procedure for sending encrypted data to the set of users that can be used by any third party (any content provider).
- Decryption.
  An algorithm that can be used in conjunction with secret key material by the user to decrypt the scrambled data and obtain plaintext.
- Traitor Tracing and Trial.
  An algorithm that can be applied by the system-manager to the information found in pirate decoder to find out the identities of the users that have revealed their keys. This algorithm should provide non-repudiable information, which can be verified in a trial by a judge.

We say that a scheme is asymmetric if it satisfies the following properties:

- frameproof: the system manager cannot frame any of the innocent users
- direct non-repudiation: tracing algorithm should produce undeniable proof of the implication of the traitors in the construction of pirate decoder. Such a proof should be verifiable for any third party (a judge) without the participation of the subscribers of the system.

3.4 The sample implementation.

The following section presents one of the possible implementations of asymmetric traitor tracing schemes introduced in [8], which as we believe is the most practical and efficient one developed as far.

3.4.1 Assumptions.

We assume that every user that wants to join the protocol possesses a digital signature mechanism, $sign_u$, which he/she can use to uniquely sign messages. There is also a corresponding verification algorithm, $verif_u$, that can be used by any third party to verify the authenticity of the signed message.

3.4.2 Initialization.

During the system initialization the system-manager selects one random polynomial in the form of: $Q_1(x) = a_0 + a_1x + \dots + a_{2v}x^{2v}$ over Z_q, and a random $b \in Z_q$ and sets $y = a_0$ and $h_0 = g$, $h_1 = g^{-a_1}, \dots, h_{2v} = g^{-a_{2v}}$, $h' = g^{-b}$. Let $Q(x,y) := Q_1(x) + by$.
The public key of the system, that can be used by any content provider to encrypt the transmitted data is published as the tuple $<y, h_0, \dots , h_{2v}, h'>$.

3.4.3 Join protocol.

To let a new user obtain a subscription service the system-manager and the user must carry out the join procedure. The goal of this procedure is to allow the new user u to compute a point $<z_u, \alpha_u, Q(z_u, \alpha_u)>$ of the polynomial Q so that: z_u is randomly selected by the system-manager and $\alpha_u = \alpha_u^C + \alpha_u^R$ where α_u^C is a values selected and committed by the user (and remains secret to the system-manager), and the value α_u^R is randomly selected by the system manager. The commitment of the user u to the value α_u^C will be of the form
$<C_u = g^{\alpha_u^C}, sign_u (C_u)>$, where $sign_u (C_u)$ means the value C_u signed with the private key of the user. The value α_u equals $\alpha_u^C + \alpha_u^R$.
The join protocol can be implemented by employing an instantiation of a OPE over a committed value.
After the completion of the Join procedure, the user's u secret personal key is the vector $K_u := <Q(z_u, \alpha_u), z_u, z_u^2, \dots , z_u^{2v}, \alpha_u>$.

At this place the following two goals are achieved, that serve the purpose of the system's asymmetry and non-repudiation: after the Join protocol the user is in possession of a valid secret-key of the system and is not known to the system-manager and the system-manager holds a non-repudiable commitment of the user to the secret portion of the user's secret key.

3.4.4 Encryption.
Any content provider can use the encryption function to distribute data to the selected set of subscribers. To do so he must obtain the public-key of the system (calculated by the system-manager during the system initialization), pk := $<y, h_0, \dots, h_{2v}, h'>$. He can then encrypt a plaintext M as follows: $<y^r \cdot M, h_0^r, \dots, h_{2v}^r, (h')^r>$ where r is a random integer less than q. So the encryption function is defined as follows:

$$E(pk, M) = <y^r \cdot M, h_0^r, \dots, h_{2v}^r, (h')^r>$$

3.4.5 Decryption.
Any ciphertext can be decrypted using a representation of y w.r.t. the base $h_0, \dots, h_{2v}, h'$. Given a ciphertext $\tilde{G}$:= $<G, G_0, G_1, \dots, G_{2v}, G'>$ and a representation of the key K := $<\delta_0, \dots, \delta_{2v}, \delta'>$ the decryption function is defined as follows:

$$D(\tilde{G}, K) := G/((G')^{\delta'} \prod_{j=0}^{2v} (Gj)^{\delta j})$$

3.4.6 Traitor Tracing.
The set of all possible decryption keys is identified with the set of all representations of y, because every representation of y w.r.t. the base $h_0, \dots, h_{2v}, h'$ can be used as a decryption key. If some of the dishonest users collude to produce a new decryption key they can make up an arbitrary representation of it, but only as a linear combinations of the secret-keys the traitors possess.

We can define following the problem to be resolved in order to achieve asymmetric traitor tracing.
Input. A vector $K = \Sigma^t_{l=1} \mu_l K_{ul}$, where $\{u_1, \dots, u_t\} \subseteq \{1, \dots, n\}$ is the set of traitor users and n is the number of all users in the system and integers $z_1, \dots, z_n$, that defined a portion of the secret-key Ku of every user.
Output. The indices $\{u_1, \dots, u_t\}$ and the values $\{\mu_1, \dots, \mu_t\}$.

The efficient algorithmic construction for the Traitor Tracing problem as presented above based on Decoding of Algebraic Codes has been introduced in [8]. The efficiency of this algorithm is $O(n^2)$ (if the Berlekamp-Welch algorithm is implemented), but more efficient implementations are possible to reduce the complexity to $O(n(\log n)^2)$. For the detailed construction of this algorithm the reader is referred to [8].

3.4.7 The Trial.
The system-manager carry out traitor tracing procedure on the pirate key
$K = <K_0, K_1, \dots, K_{2v}, K'>$ with $K = \Sigma^t_{l=1} \mu_l K_{ul}$ where $\{u_1, \dots, u_t\} \subseteq \{1, \dots, n\}$ is the set of traitor users and obtains as output the vector $v = <v_1, \dots, v_n>$ where $v_{ul} = u_l$ for $l = 1, \dots, t$ and $v_1 = 0$ for all $i \in \{1, \dots, n\} - \{u_1, \dots, u_t\}$. He must afterwards convince the judge of the implication of the users $\{u_1, \dots, u_t\}$ into construction of the pirate key K. This is performed in the following way: the system-manager presents to the judge the vector v, the pirate key K, the values $\alpha_{u1}^R, \dots, \alpha_{ut}^R$ and the commitments of the suspected users $C_{u1};sign_{u1}(C_{u1}), \dots, C_{ut};sign_{ut}(C_{ut})$ (generated during the join procedure). The judge first verifies all signatures by applying the verification algorithms
$verif_{u1}, \dots, verif_{ut}$ on them. Afterwards he verifies if the following equation is true:

$$\prod_{l=1}^{t} (C_{ul} g^{\alpha_{u1}^R})\, v_{ul} = g^{K'}$$

If the equation holds the judge concludes that indeed the users $\{u_1, \dots, u_t\}$ were implicated in the construction of the pirate device with the key K. As we can notice the direct non-repudiation is obtained: any third party can obtain a proof of the traitors' implication in the treachery using the information provided by the tracer without the direct participation of the suspected users.

4. Summary and Outlook

In our assignment we have presented traitor tracing as one of the mechanisms that can be applied to prevent illegal digital content distribution. The idea behind traitor tracing algorithms is to deter the legal subscribers form leaking their key information to the pirates, as it will prevent the creation of pirate decoders. For that reason traitor tracing can be a powerful tool (especially in conjunction with fingerprinting methods) for tracing the dishonest users and punishing them for the treachery. However, the most popular and best developed symmetric traitor tracing algorithms have, in our opinion, a major drawback: they cannot be used for obtaining legal evidence that can convince any third party (a judge) of the implication of the suspected user in the construction of pirate device. The existence of such evidence would allow the system-manager to undertake some legal actions against the traitors, thus making the treachery less attractive and more dangerous. The asymmetric traitor tracing schemes belong, unfortunately, to the least developed and researched area, although some interesting publications presenting more practical and convincing implementations of the asymmetric algorithms have appeared recently. We have presented in our assignment one of them as an example of how such a problem can be solved theoretically. For the time being we do not know about any real life implementation of such solutions, but the economical need for digital rights protection will most probably force the deployment of similar schemes in the future.

References

[1] Benny Chor, Amos Fiat and Moni Naor. Tracing traitors. Advances in Cryptology – Crypto '94, Lecture notes in Computer Science 839, (1994), pp. 257-270.

[2] D.Boneh and M. Franklin. An efficient public key traitor tracing scheme. Advances in Cryptology – Crypto '99.

[3] Dwork, Lotspiech, and Naor "Digital Signets: Self-enforcing protection of digital information" in 28 th Symp Theory of Computation, 1996, pp 489-498.

[4] D.Boneh and J. Shaw "Collusion-secure fingerprinting for digital date" IEEE Trans. Inform. Theory, vol. 44, pp 1897-1905, Sept 1998

[5] A.Fiat and T. Tassa, "Dynamic Traitor Tracing" in Proc Advances in Cryptology-Crypto '99, 1999, pp. 388-397.

[6] Birgit Pfitzmann, "Trials of Traced Traitors", Information Hiding Workshop, Spring LNCS 1174, pp. 49-63, 1996

[7] K.Kurosawa and Y.Desmedt, Optimum Traitor Tracing and Asymmetric Schemes, Eurocrypt 1998

[8] A. Kiayias and M. Yung, Breaking and Repairing Asymmetric Public-Key Traitor Tracing, DRM 2002

[9] H. Komaki,Y. Watanabe, G. Hanaoka and H. Imai, Efficient Asymmetric Self-Enforcement Scheme with Public Traceability, Public Key Cryptography 2001

[10] D.R. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes." SIAM vol11, pp 41-53, 1998

[11] J.N. Staddon, "A combinational study of communication, storage and traceability in broadcast encryption systems" Ph.D University California, Berkley 1997

[12] Y. Watanabe, G. Hanaoka and H. Imai, Efficient Asymmetric Public-Key Traitor Tracing without Trusted Agents, CT-RSA 2001