Traitor Tracing through Function Sharing

by

Cem Çelebiler

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Engineering

and

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1996

Author .....
Department of Electrical Engineering and Computer Science
December 15, 1995

Certified by .....
Silvio Micali
Professor of Computer Science
Thesis Supervisor

Accepted by .....
F.R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Traitor Tracing through Function Sharing

by

Cem Çelebiler

Submitted to the Department of Electrical Engineering and Computer Science
on December 15, 1995, in partial fulfillment of the
requirements for the degrees of
Master of Engineering
and
Bachelor of Science in Computer Science and Engineering

Abstract

To broadcast information (for example pay-television) over insecure channels (for example the public airwaves) to a set of paying subscribers, the broadcaster may employ encryption to prevent non-subscribers from gaining access to the programming. A risk to the broadcaster is that subscribers may sell or give their decryption key to non-subscribers and thus deprive the broadcaster of potential revenue. Traitor tracing schemes are key management schemes that allow the broadcaster to trace any subscriber who sells his key.

Chor, Fiat and Naor introduced the concept of traitor tracing schemes and generalized it to schemes that ensure traceability of traitors as long as the number of traitors is bounded by some number k. They identified the following performance parameters,

1. Memory requirements of the data supplier.
2. Memory requirements of a user
3. Size of broadcast overhead to establish a session key

We propose a traitor tracing scheme that can be based on any function sharing (or threshold function) scheme that satisfies an additional technical property which we define.

A number of threshold function schemes based on various different cryptographic primitives have been proposed. A particularly efficient implementation of threshold function schemes is possible using the ElGamal function. We conjecture that this ElGamal based threshold function scheme has the required additional technical property. A version of our traitor tracing scheme implemented on top of the ElGamal based threshold function scheme is more efficient than the Chor, Fiat, Naor scheme in terms of both broadcast overhead and memory requirements per user. On the other hand, our scheme has higher computation costs.

Thesis Supervisor: Silvio Micali
Title: Professor of Computer Science

Acknowledgments

I would like to thank my thesis supervisor, Silvio Micali, for helping me vastly improve the exposition of my rather vague initial ideas. I am very grateful for the time he generously gave to read and re-read drafts of my thesis, and the numerous hints he offered on how to express the key ideas clearly and succinctly and where to trim unnecessary clutter.

Paul Sauer has made innumerable contributions to this thesis and to my understanding of mathematics among other things. Without his enthousiasm for this problem, his infusion of ideas, his understated criticism and his general refusal to consider the possibility of defeat I would have probably given up long ago.

I would like to thank my parents for supporting me and my brothers Yunus and Nedim for much encouragement,

I would like to thank Tony Eng, Rosario Gennaro, Adi Shamir, Stefan Brands and Scott Decatur for helpful discussions.

Finally, and most importantly, I would like to thank Emily Dorsett for her kindness, encouragement and support.

Chapter 1

Introduction

Suppose a data supplier wants to broadcast information such as television programming or news reports to a large set of paying subscribers. If the broadcast medium is insecure (for example the public airwaves), the broadcast information must be encrypted to prevent non-subscribers from gaining access to it. To allow the subscribers to access the data, the decryption key is given to each of the subscribers.

What happens, however, if a malicious subscriber (called a *traitor*) wants to give or sell access to the data to non-subscribers? Clearly, a traitor can simply sell the plaintext to non-subscribers, either through direct rebroadcast or through storage and later resale. There is no cryptographic means of preventing this from happening. In practice, however, this means of *piracy* is generally not viable. Often the broadcast information loses value over time (e.g. sports events, newspaper headlines) and therefore storage for later resale is not worthwhile. Rebroadcasting is typically prohibitively expensive, and if it is conducted on a non-negligeable scale it can be traced through electromagnetic detection equipment (e.g. the equipment that police use to locate pirate radio broadcasters).

It is much more likely, therefore, that in practice the traitor will simply give or sell the decryption key to non-subscribers to allow them to decrypt the broadcast. It is well known how to prevent this form of piracy in an interactive setting, but it is less obvious how it can be prevented in a purely broadcast setting where there is no return communication path and the service provider broadcasts the same information to subscribers and non-subscribers alike. In this paper, we provide effective mechanisms for discouraging this form of piracy in a non-interactive broadcast setting.

An obvious way of preventing a traitor from selling his key is to hide the key from the subscribers. Unfortunately, the only means of doing this is through the use of so-called tamper-proof (or secure) hardware. Supposedly it is not possible to learn the value of a key embedded in tamper-proof hardware. It is not clear, however, how well these devices really work, and we have dismissed this option as a primary means of security. We note, however, that using secure hardware can only improve the security of a scheme, and therefore in practice it may be used as an additional "shield".

An alternative to preventing malicious subscribers from building pirate decoders is to ensure traceability of traitors by assigning each user a distinct *personal key*. This

means of discouraging piracy, "Tracing Traitors" was introduced in [3]. The idea behind tracing traitors is to distribute different keys to each user, thereby allowing one to identify traitors *after the fact* by comparing the keys of confiscated pirate decoder boxes with the list of subscribers' keys. This should act as a disincentive to malicious subscribers. In this paper, we present a novel traitor tracing scheme.

A trivial traitor tracing scheme consists in encrypting the data separately with each personal key. In practice, the data would be encrypted with a *session-key* s, and the data supplier would broadcast the encryption of s with each of the personal keys. Periodically the data supplier would change the session-key and again broadcast its encryption using each of the personal keys. We denote the amount of data which the data supplier must broadcast to establish a session-key *broadcast overhead.*

Unfortunately, the broadcast overhead of this trivial traitor tracing scheme is prohibitive: an encrypted key must be broadcast for each of the n subscribers. If there are many subscribers, so much bandwidth is spent on broadcasting the encryptions of the session-key that it crowds out the actual information (TV shows, etc) being broadcast.

In order to reduce the size of the broadcast overhead we may consider trading off some degree of traceability. In "Tracing Traitors" [3], Chor, Fiat and Naor propose schemes that allow one to trace at least one traitor as long as the number of collaborating traitors is k or fewer. They identify the following performance parameters

1. Memory and computation requirements for the data supplier.
2. Memory and computation requirements for a subscriber
3. Size of the broadcast overhead needed to establish a session-key.

THE CHOR, FIAT, NAOR SCHEMES

Chor, Fiat and Naor propose k-traitor tracing schemes using combinatorics based on top of any underlying encryption routine whose keys we will call *elementary keys.*

In [3], each user holds a collection of elementary keys from the underlying encryption scheme. They assign these elementary keys to users as follows. First they consider an $l \times m$ array of elementary keys

$S(1,1)$	...	$S(1,m)$
$\vdots$	$\vdots$	$\vdots$
$S(l,1)$	...	$S(l,m)$

and a matching collection of l hash functions, $h_1, \ldots, h_l$ hashing from the set of all subscribers to $\{1, \ldots, m\}$. They then distribute to each subscriber u a collection of l elementary keys: one key per row, $S(1, h_1(u)), \ldots, S(l, h_l(u))$.

To establish a session-key s, the broadcaster chooses l values $s_1, \ldots, s_l$ such that the bitwise XOR of the s_i's yields s ($\bigoplus_i s_i = s$), and broadcasts the encryption of each s_i using each of $S(i,1), \ldots, S(i,m)$. The broadcast overhead is thus $l \times m$ elementary keys. For each row i, each subscriber u holds a key $S(i, h_i(u))$ and can hence obtain

$s_1, \ldots, s_m$ and thereby compute s. If l and m are chosen appropriately (they are functions of the number of subscribers n and the maximum coalition size k), then it is possible to show that there exist hash functions $h_1, \ldots, h_l$ such that any pirate decoder box built by a coalition of k or fewer traitors is traceable to at least one of the traitors.

EFFICIENCY OF THE CHOR, FIAT AND NAOR SCHEMES

For n subscribers and k or fewer traitors Chor, Fiat and Naor establish values of l and m that result in each user holding a collection of $O(k^2 \log n)$ elementary keys and a broadcast overhead consisting of $O(k^4 \log n)$ (encrypted) elementary keys. The size of each elementary key is the key size of the underlying encryption scheme used. In practice a secret key encryption scheme uses keys of size around 100 bits. In particular one could use a 64 bit DES key.

They propose a second scheme (using two "levels" of hash functions) that trades off number of keys per user against broadcast overhead. In this second scheme each user stores $O(k^2 \log^2 k \log n)$ elementary keys and the broadcast overhead consists of $O(k^3 \log^4 k \log n)$ (encrypted) elementary keys. Finally, they also present a probabilistic "secret" scheme [1] were the traitors' probability of constructing an untraceable decoder is a parameter p_t. In this scheme each user stores $O(k \log(n/p_t))$ elementary keys and the broadcast overhead consists of $O(k^2 \log(n/p_t))$ elementary keys [2]. All the schemes in [3] are nonconstructive. The constructive variants they propose are considerably less efficient.

1.1 Scope of this thesis

The notion of traceability, like the notion of pseudo-randomness, elicits an intuitive understanding yet is difficult to define precisely. Before the notion of pseudo-randomness was fully formalized, researchers presented schemes that satisfied different requirements and operated under different assumptions. Often the underlying assumptions were not clearly identified. The notion of traceability is currently in that state of understanding.

Eventually, after a long gestation period, the notion of pseudo-randomness was clearly defined and the necessary assumptions were distilled to a single, purely mathematical assumption [1, 15]. Traceability has not yet attained that stage of formalization. It is the author's intent to work towards a clean formalization of this notion which reduces the necessary requirements to a minimal mathematical assumption. In the meantime, however, I believe that it is still worthwhile to put forward a promising idea for a traceability scheme, even in the absence of a completely formalized frame-

[1] The "secret" is the set of hash functions. They must remain secret because an adversary knowing them could choose which k subscribers to corrupt and defeat the traceability of the scheme

[2] The authors of [3] have informed me that since the publication of that article they have been able to improve the efficiency of their schemes by a factor of k. In particular their probabilistic "secret" scheme can be shown to require each user to store only $O(\log(n/p_t))$ elementary keys and the broadcast overhead consists of $O(k \log(n/p_t))$ elementary keys.

work. This thesis proposes a new idea for a traceability scheme while attempting to clarify the notion of traceability.

1.2 Our contribution

In this thesis we propose a k-traitor tracing scheme that is based on threshold function (or function sharing) schemes [4, 5]. The traceability of our scheme relies on using the shares of the threshold scheme to identify the subscribers. This use of shares of threshold schemes for identification is novel and requires an additional property of the shares that is not required by the simple definition of threshold schemes: namely that the shares cannot be disguised. We develop and define this notion which we call share traceability.

Our traitor tracing scheme can be based on any non-interactive threshold function scheme that satisfies this property. As in [3], we assume that the contents of any pirate decoder are visible to the tracing algorithm. Each user is required to store one share function, and the broadcast overhead consists of the images of k share functions at some point.

There are many *candidate* implementations of threshold function schemes that could be used in our traitor tracing scheme. In particular we conjecture that a particularly efficient implementation of threshold function schemes using the ElGamal function satisfies the share traceability property. The ElGamal function [7] is derived from the Diffie-Hellman key exchange protocol [6] and is similarly based on the Discrete Log Problem. To perform function sharing we use ElGamal moduli of the form $p = 2mq + 1$, where p and q are large primes ($|p| > 1000$ bits and $|q| > 160$ bits) and m is an arbitrary integer. Moduli of this form are fairly commonly used in cryptography, notably in Schnorr's signature scheme [14]. An ElGamal version of our traitor tracing scheme achieves broadcast overhead $k \cdot |q|$ while requiring each user to store $|q|$ bits, an improvement in both measures over [3]. In addition, the schemes we propose are constructive whereas the [3] schemes are not. On the other hand, the computation costs are higher in our scheme: $O(k)$ exponentiations (with exponents of length $|q|$) are necessary whereas the [3] schemes need only $O(k \log n)$ XOR operations.

Unfortunately, while we conjecture that many known threshold function schemes, including the ElGamal scheme, satisfy the traceable shares property, we don't know how to prove it. As evidence for this conjecture, we claim that the ElGamal scheme possesses a related but slightly weaker property. This weaker property is interesting in its own right because it corresponds to achieving traceability in a practical setting commonly used in satellite television: keys are stored on smart cards that interact with keyless decoder boxes which execute a specific algorithm [13]. In this setting, piracy of smart cards is permitted but the decoder boxes are presumed to be unpiratable. The rational for this distinction is presumably based on the argument that the clandestine distribution of smart cards is feasible because of their small size and cost, while it is not for bulky and expensive to manufacture decoder boxes. In this setting, the ElGamal based traceability scheme we propose can be modified to delegate most

of the computational load to the decoder, while the smart card need only perform one exponentiation.

1.3 Related Work

Boneh and Shaw take up a similar problem, fingerprinting of data, in [2] and describe a scheme very similar to the Chor, Fiat, Naor scheme.

In "Broadcast Encryption" [9], Fiat and Naor consider the problem of efficiently broadcasting to any subset of a (large) set of subscribers such that only subscribers in the subset learn the message. This is a different problem, but can be used in conjunction with a traitor tracing scheme to simplify over-the-air disabling of identified pirate decoder boxes.

1.4 Applications

An obvious practical instance of this problem is pay satellite television broadcasting. A broadcaster wants to broadcast programming via satellite and sell subscriptions in the form of decoder boxes (or smart cards that plug into decoder boxes). As noted above, a subscriber may be able to open his decoder box (smart card), extract the key, and manufacture and sell pirate decoder boxes (smart cards). If the broadcaster is able to get access to a pirate decoder box (for example by purchasing one), he should be able to identify the treacherous subscriber. Note that once the broadcaster has traced the traitor, he can disable the pirate boxes either efficiently by using a broadcast encryption scheme or by rebroadcasting new personal keys to each non-treacherous user using their current personal key. This latter option is inefficient ($O(n)$ transmission) but should occur rarely enough to be a viable option.

A similar application is Global Positioning System (GPS) equipment. The United States Government has an array of satellites that continuously broadcast positioning information that can be picked up and triangulated by openly available equipment to produce latitude and longitude. For military security reasons, the satellites broadcast two sets of data: one unencrypted low precision set that can be used for triangulation to within tens of meters and another encrypted high precision set for exclusive military use. Now if the military uses a traitor tracing scheme in its equipment, it will be able to determine which GPS unit was the source of the leak when it is determined that the code has fallen in enemy hands. In addition the military can then disable the offending device over the air as mentioned above.

Fingerprinting and CD-ROMS as described in [2], [3].

For concreteness, in this thesis we will discuss the problem in the context of satellite television, and will use its terminology. The scheme is applicable to any of the other applications mentioned above, and the reader should have no trouble in making the translation.

1.5 Organization of the thesis

First we define and discuss the notions of traceability schemes and threshold function schemes. We introduce a property of threshold schemes, share function traceability, and discuss how it is related to traceability schemes. In the following chapter we present a traceability scheme that can be based on any threshold function scheme that satisfies the traceable share function property. Finally we put forward an efficient traceability scheme based on an ElGamal implementation of threshold function schemes. We give evidence that this threshold possesses traceable share functions property by claiming that it possesses a related, slightly weaker, property.

Chapter 2

Definitions and Model

In this chapter we define and discuss the notions of traceability and threshold function schemes. We define the notion of traceability and introduce the notion of traceable share functions. For simplicity, we specify most of our definitions in terms of absolute possibilities and impossibilities rather that in terms of bounded probabilities. These definitions have simple probabilistic counterparts.

2.1 Traceability Scheme

In a *traceability scheme* we distinguish the following components,

- A *user initialization scheme*, used by the data supplier to assign personal keys to users.
- A *session-key encryption scheme*, used by the data supplier to encrypt a new session-key. We call the output of this scheme, which is broadcast, the *enabling block.*
- A *session-key decryption scheme*, used by every user to decrypt the session-key from the enabling block and from the user's personal key.
- A *traitor tracing algorithm*, used upon confiscation of a pirate decoder, to determine the identity of a traitor.
- A bound β on the number of times the session-key encryption scheme can be employed.

Definition 1 *A pirate decoder is any device which on input an enabling block, outputs the corresponding session-key.*

Definition 2 *An n-user traceability scheme is called k-resilient if for any pirate decoder built by k or fewer traitors, the traitor tracing algorithm identifies one of the traitors.*

The most crucial component of a traceability scheme is clearly the traitor tracing algorithm. Unfortunately this component is also the most vague. What are its inputs? How may it interact with a confiscated pirate decoder?

In [3], Chor, Fiat and Naor adopt the assumption, which we will call *Transparent-Box Assumption*, that the traitor tracing algorithm has access to the full contents, including both the decrypting algorithm and any keys, of the confiscated pirate decoder. We call a traceability scheme that successfully traces under this assumption, *transparent-box traceable.*

Definition 3 *A k-resilient traceability scheme is called transparent-box traceable if on input the contents of any pirate decoder built by k or fewer traitors, the traitor tracing algorithm identifies one of the traitors.*

The transparent-box assumption is a fairly strong and unrealistic assumption because traitors may try to hide their keys through the use of secure hardware. Nevertheless, it is a good starting point for examining the traceability of a scheme. Eventually, we would like to strengthen the notion of traceability, by requiring that the traitor tracing algorithm be able to identify a traitor simply by interacting with a pirate decoder in a black-box fashion. Properly formalizing this notion of *black-box resiliency* is difficult because assumptions must be made as to how a pirate decoder may behave. As an example of the difficulties involved, note that a pirate decoder may simply self-destruct as soon as it notices an input that looks different from the standard over the air broadcast. Chor, Fiat, and Naor parenthetically make the claim in [3] (without proof or sketch) that their schemes are black-box resilient, but as they do not specify a model of how the pirate decoder may operate, it is difficult to interpret their claim.

Having defined what me mean by a traceability scheme, we now look at a cryptographic primitive, threshold function schemes, to see how they can be used to construct a traceability scheme. We will introduce a property of threshold function schemes, share traceability. In the next chapter we will then present a transparent-box traceability scheme based on any traceable threshold function scheme.

2.2 Threshold Function Schemes

Our scheme relies on a recent cryptographic primitive known as function sharing or threshold function schemes. The idea of this primitive is to split a hard to compute trapdoor function f into share functions (or shadow functions) with the following two properties

Definition 4 *A (t, l) threshold function scheme for a trapdoor function f consists of a pair of probabilistic polynomial time (PPT) algorithms* FUNCSHARE *and* FUNCREC. FUNCSHARE *takes as input the number of shares to generate, l, and the threshold value, t, and outputs l share functions $P_1, \ldots, P_l$.* FUNCREC *takes as input any element w from the range of f and any t share function evaluations at w,*

*$P_{i_1}(w) \ldots, P_{i_t}(w)$, and outputs an element u of the domain of f. The threshold function scheme (*FUNCSHARE,FUNCREC*) is called (t, l)-secure if the following two properties hold.*

1. *For any u in the domain $\mathcal{D}_f$ of f, on input $w = f(u)$, the value of t share functions at w, $P_{i_1}(w) \ldots, P_{i_t}(w)$,* FUNCREC *outputs u.*

2. *The trapdoor function f remains hard to invert for all values w at which one knows the image of less than t share-functions, even after having seen the value of t share-functions at any number of points other than w. More formally, we define this property as follows. Let $\Gamma = \{1, \ldots, l\}$. Let $\mathcal{R}_f$ be the range of f. For all $\{i_1, \ldots, i_j\} \in \Gamma$, where $0 < j < t$, for all probabilistic polynomial time algorithms $\mathcal{A}$, for any polynomial poly$(\cdot)$,*

$$\Pr[f(u) = w : (P_1, \ldots, P_l) \leftarrow \text{FUNCSHARE}(t, l); \\ w \in_R \mathcal{R}_f; u \leftarrow \mathcal{A}(w, H, P_{i_1}, \ldots, P_{i_j})] < \frac{1}{poly(|\mathcal{D}_f|)}$$

where H is a history tape of length polynomial in $|\mathcal{D}_f|$ whose m^{th} entry contains a random element α_M of the domain of f, its image $f(\alpha_m)$ and at least t share function evaluations corresponding to the index set $\Lambda_m \in \Gamma(|\Lambda_m| \geq t)$. More succinctly, it is the tuple $(\alpha_m, f(\alpha_m), P_{i_{m,1}}(\alpha_m), \ldots, P_{i_{m,|\Lambda_m|}}(\alpha_m))$. (Note that $f(\alpha_m)$ is included, but can be omitted wlog as it can be computed given the $P_{i_{m,j}}(\alpha_m)$'s in polynomial time).

In the traitor tracing schemes we propose, each user will hold a share function. The traceability of our schemes rely on the uniqueness of each share function. The idea is that any pirate decoder built by $t-1$ or fewer traitors will contain at least one of the traitors' share functions and therefore it will be traceable to that traitor. Unfortunately, nothing in the above definition of threshold function schemes precludes the possibility of disguising a share function in such a way that even with the transparent-box assumption it is not possible to relate the contents of a pirate decoder to any of the subscribers' share functions. Indeed the rich structure of threshold function shares appears to render them particularly vulnerable to being disguised.

Although the [3] schemes have keys that do not have such a rich structure, nonetheless it is possible that they too can be disguised in some way. The authors of [3] ignore this possibility and there is no provision or proof in their scheme that precludes the possibility of traitors disguising their keys. We believe that this attack must be considered in any traitor tracing scheme.

In order to prevent the possibility of a disguising attack, we require that the threshold function scheme we use in our traitor tracing scheme have shares that cannot be disguised. We state the property we require a bit more formally using some helpful definitions as follows.

Definition 5 *A share-disguising algorithm for a (t, l) threshold function scheme is an algorithm that on input a history tape and $t-1$ share functions outputs a collection of disguised shares $\tilde{\mathcal{D}}$.*

Definition 6 *A pirate-reconstructing algorithm for a (t, l) threshold function scheme is an algorithm that on input a collection of $t-1$ or fewer possibly disguised shares $\tilde{\mathcal{D}}$, any w in the range of f, and the values of any $t-1$ share functions at w, outputs an element u of the domain of f as a guess at the inverse of w.*

Definition 7 *A share-extracting algorithm for a (t, l) threshold function scheme is an algorithm that takes as input a collection of $t-1$ or fewer possibly disguised share functions, and outputs an element of the space of share functions.*

Property 1 (Traceable Share Functions) *A threshold function scheme with threshold value t has traceable share functions if*
For all PPT pirate-reconstructing algorithms PIRATEREC, *there exists a PPT share-extracting algorithm* EXTRACT *such that for all collections of $t-1$ share functions $P_{e_1}, \ldots, P_{e_{t-1}}$, for all PPT share-disguising algorithms* DISGUISE, *for all l large enough and for all share functions $\{P_{i_1}, \ldots, P_{i_{t-1}}\} \subset$ FUNCSHARE(t, l) disjoint from $\{P_{e_1}, \ldots, P_{e_{t-1}}\}$, whenever the following hold,*

- $\tilde{\mathcal{D}} =$ DISGUISE$(H, P_{e_1}, \ldots, P_{e_{t-1}})$ *where H is a history tape as defined in definition 4 and*
- $\forall w = f(u)$ *in the range of f,* PIRATEREC$(\tilde{\mathcal{D}}, w, P_{i_1}(w), \ldots, P_{i_{t-1}}(w)) = u$

then EXTRACT$(\tilde{\mathcal{D}}) \in \{P_{e_1}, \ldots, P_{e_{t-1}}\}$

Chapter 3

Transparent-Box Traceability Scheme

In this chapter we present a transparent-box traceability scheme that can be based on any share traceable threshold function scheme.

3.1 User Initialization Scheme

The broadcaster will generate the l share functions $P_1, \ldots, P_l$ of a share traceable (t, l) function sharing scheme, choosing $l = n + \beta$ and $t = k + 1$. The broadcaster will then give user i $(1 \leq i \leq n)$, share function P_i and will retain the β share-functions $P_{n+1}, \ldots, P_{n+\beta}$.

3.2 Session-Key Encryption Scheme

When the broadcaster wants to establish a new session-key, he chooses k elements $c_1, \ldots, c_k$ each at random from $\{n+1, \ldots, n+\beta\}$. He then chooses a random element u from the domain of f as the session-key and computes $w = f(u)$. The broadcaster then broadcasts

$$w, P_{c_1}(w), P_{c_2}(w), \ldots, P_{c_k}(w)$$

that is w as well as the value of k $(= t - 1)$ share functions at w.

3.2.1 Session-Key Decryption Scheme

Any subscriber i can simply compute $P_i(w)$, combine it with the $t - 1$ other share values at w, $P_{c_1}(w), \ldots, P_{c_k}(w)$, that are broadcast, and by virtue of property one of function sharing, compute the session-key, u.

3.3 Traitor Tracing Algorithm

Without loss of generality we view the contents of the pirate decoder as consisting of a decrypting algorithm DECRYPT and a set of keys $\tilde{\mathcal{D}}$. Now notice that the inputs to the decrypting algorithm are the same as those of a pirate reconstructing algorithm: a set of possible disguised keys put together by $t-1$ or fewer traitors, a point in the domain of f and $t-1$ share function evaluations at that point. Therefore we are in the context of the Traceable Share Functions Property, and can consider DECRYPT to be a pirate reconstructing algorithm.

Now by the share function traceability property of the underlying threshold scheme, there exists an extractor EXTRACT for DECRYPT. We run EXTRACT with the set of keys as input and output the identity of the subscriber that corresponds to the share function that EXTRACT outputs.

If the output of this traitor tracing algorithm is not the identity of one of the traitors then we have violated the traceable shares property of the underlying threshold function scheme.

3.4 Conclusion

We have have presented a transparent-box traceability scheme and argued that it meets the definition. Unfortunately we do not know of any threshold function scheme for which we can prove that it has the traceable share function property. In the next section, we will introduce a weaker notion of share traceability, and claim that an ElGamal based threshold function scheme possesses this weaker property. This weaker property translates into a level of security that corresponds to the security of current smart card satellite television schemes.

Chapter 4

ElGamal based scheme

An efficient ElGamal based implementation of non interactive threshold function schemes has been proposed in [5] using a technical construction already used in [8] in a different context. Unfortunately we don't know how to prove that it satisfies the traceable share functions property.

Conjecture 1 *The ElGamal based threshold function scheme as defined in [5] has traceable share functions*

As evidence for this conjecture, we claim that it satisifies a weaker notion of share traceability: rather than claiming that for any pirate-reconstructing algorithm there exists a share extractor, we claim that there exists a specific reconstructing algorithm for which there is an extractor. In fact, the reconstructing algorithm itself learns the identity (but not the secret share) corresponding to the share function evaluation presented to it as input.

Property 2 *We say a threshold function scheme has a share enforcing reconstruction algorithm if there exists a reconstructing algorithm* FUNCREC *that satisfies*
For all $t-1$ *pirate shares* $P_{e_1}, \ldots, P_{e_{t-1}}$, *for all disguising algorithms* DISGUISE, *for all share functions* $\{P_{i_1}, \ldots, P_{i_{t-1}}\}$ *disjoint from* $\{P_{e_1}, \ldots, P_{e_{t-1}}\}$, *if*

- $\tilde{\mathcal{D}} = \text{DISGUISE}(P_{e_1}, \ldots, P_{e_{t-1}})$, *and*
- $\forall w = f(u)$ *in the range of* f, $\text{FUNCREC}(w, P_{i_1}(w), \ldots, P_{i_{t-1}}(w), \tilde{\mathcal{D}}(w)) = u$

then either

- $\tilde{\mathcal{D}}(w) \in \{P_{e_1}(w), \ldots, P_{e_{t-1}}(w)\}$ *or*
- $\text{FUNCREC}(w, P_{e_1}(w), \ldots, P_{e_{t-1}}(w), \tilde{\mathcal{D}}(w)) = u$

In other words, in order for the share enforcing reconstructing algorithm to work, the input provided by the traitors must either be the evaluation of one of their share functions (which is traceable back to them), or they must have broken the threshold function scheme.

Theorem 1 *The ElGamal based threshold function scheme as presented in [5] has a share enforcing reconstruction algorithm.*

The proof of this theorem is not given in this thesis but will be given in a subsequent paper.

Aside from its interest as evidence for conjecture 1, this property suffices to ensure traceability in the model of traitor tracing commonly used in practical satellite television schemes. This model specifies that keys reside on smart cards that interact with keyless decrypters that execute a common algorithm [13, 10]. In this model, the only permissible piracy is the manufacture of pirate smart cards that work in the fixed decrypters (i.e. manufacture of pirate decrypters is not permitted). We adapt our scheme to this model by simply embedding the share enforcing reconstruction algorithm in the decrypter. By theorem 1 then, the decrypter learns the identity of any smart card interacting with it and hence manufacture of untraceable pirate smart cards is not possible.

As we noted, this model of piracy is commonly (implicitly) adopted in practice. The way in which the model is used is quite different however. A common current satellite television decoder system, Videocrypt, separates the decrypting operations from the identification operations of the smart card [12, 11]. In that scheme, the smart card contains two sets of keys: a decryption key, shared by all smart cards, and a Fiat-Shamir identification private-public key pair unique to each smart card. The decrypter uses the decryption function of the smart card to decrypt session keys and periodically also engages in an identification protocol with the smart card to verify its identity. We believe that our scheme is superior to such a scheme because in our scheme decryption and identification are closely linked: whenever the smart card enables the decrypter to decrypt, it identifies itself.

Bibliography

[1] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[2] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. In *Advances in Cryptology-Proceedings of Crypto '95*, pages 452–465, Berlin. Springer-Verlag.

[3] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Advances in Cryptology-Proceedings of Crypto '94*, pages 257–270, Berlin. Springer-Verlag.

[4] Alfredo de Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 522–533, 1994.

[5] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology-Proceedings of Crypto '89*, pages 307–315, Berlin. Springer-Verlag.

[6] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[7] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[8] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.

[9] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology-Proceedings of Crypto '93*, pages 480–491, Berlin. Springer-Verlag.

[10] Louis C Guillou. Smart cards and conditional access. In *Advances in Cryptology-Proceedings of Crypto '85*, pages 480–489, Berlin. Springer-Verlag.

[11] G. Hashkes and Cohen M. Managing smart card for pay television: the videocrypt approach. In *ACSA 90*, page 213.

[12] John McCormac. *European Scrambling Systems: Circuits, Tactics and Techniques.* Waterford University press, 1993.

[13] Patrice J.Y. Peyret. Defeating pay-tv pirates with smart cards. *IEEE Transactions on Consumer Electronics*, 20:316–317, 1990.

[14] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[15] A.C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.