Digital Signets: Self-Enforcing Protection of Digital Information
(Preliminary Version)

Cynthia Dwork* Jeffrey Lotspiech* Moni Naor†

Abstract

The problem of protecting digital content – software, video, documents, music, *etc.* – from illegal redistribution by an authorized user, is the focus of considerable industrial and academic effort. In the absence of special-purpose tamper-proof hardware, the problem has no cryptographically secure solution: once a legitimate user has purchased the content, the user, by definition, has access to the material and can therefore capture it and redistribute it. A number of techniques have been suggested or are currently employed to make redistribution either inconvenient or traceable. In this paper we introduce *digital signets*, a new technique for protecting digital content from illegal redistribution.

The work motivates the study of the previously unexamined class of *incompressible* functions, analysis of which adds a cryptographic twist to communication complexity.

1 Introduction

The problem of protecting mass-distributed digital content, such as software, audio, video, and digital library data, from illegal redistribution by an authorized user, is the focus of considerable industrial and academic effort [2, 5, 6, 10, 8, 12, 15, 20]. In the absence of special-purpose tamper-proof hardware, the problem has no cryptographically secure solution: once a legitimate user has purchased the content, the user, by definition, has access to the material. Nothing but inconvenience prevents her from selling or redistributing the decrypted version. The inconvenience comes from the *amount* of data to be redistributed, i.e. fighting piracy by bandwidth. This is particularly relevant to large software packages, videos, and anything else whose volume is on the order of the storage capacity of a CD-ROM, since these are generally too large to distribute conveniently by e-mail or by diskette and most people do not own read/write CD-ROM drives. We are therefore interested in a scheme in which the decryption key is very long. Theoretically, it should be as long as the program itself.

A number of techniques have been suggested or are currently employed to make redistribution either inconvenient or traceable. The problem with relying on traceability is that it requires a digital content police force to monitor the world for potentially stolen content, with no automatic means for generating suspicion. In this paper we introduce *digital signets*, a new technique for protecting digital content from illegal redistribution, whose goal is to motivate the user to be self-policing. In a digital signet scheme, a pirate attempting to illegally redistribute the content is given a choice between exactly two alternatives: either reveal some sensitive information (such as a credit card number) or use a channel of essentially the same bandwidth as the original distribution channel.

In broad terms, signets work as follows. There is some common public data, some of which is encrypted *content*, an *authorization center*, and any number of potential *users*. Each user has access to the common public data. To gain access to the content, user u interacts with an authorization center to obtain a short *digital signet*, which is a function of information private to u. Using only the public data, its own private information, and the signet, u can decrypt the content. Signets are designed in such a way that for u to help any other u' to access the content, u would have to either transmit something very long (such as the content itself), or reveal u's private information. Thus the contributions of this work are the *concept* of self-policing via sensitive information and a *method* for embedding sensitive data into keys.

For concreteness we focus on a scheme for distributing software via CD-ROM. Other applicable scenarios include on-line services and databases, such as patent or legal databases, newspapers, and journals, and pay TV. (See Section 8 for further discussion.) Thus, we assume that the (encrypted) software and some additional (unencrypted) data are stored on a CD-ROM, which is distributed for free. Note that the copies are identical, as the CD-ROMs cannot be personalized while retaining economy of scale. A good

*IBM Almaden Research Center, 650 Harry Road, San Jose CA 95120, USA. Research supported in part by the US-Israel Binational Science Foundation. E-mail: {dwork,lotspiech}@almaden.ibm.com.

†Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by grants from the Israel Science Foundation administered by the Israeli Academy of Sciences and by the US-Israel Binational Science Foundation. Part of this work was done while the author was visiting the IBM Almaden Research Center E-mail: naor@wisdom.weizmann.ac.il.

choice for the private information is a credit card number, although any "sensitive" information that the user does not wish to compromise will do.

The pair consisting of the user's sensitive information and the corresponding signet is called a *signet pair.* The CD-ROM contains an unencrypted *extrication* program that, on input a valid signet pair, extracts a decryption key, which is then used to decrypt the encrypted software. Note that the user's credit card number or other sensitive information is trivially obtained from the signet pair.

Ideally, to prevent a dishonest user from acting as a pirate authorization center, signets should have a security property analogous to existential unforgeability of a digital signature scheme against an adaptive chosen message attack as defined in [18]. In such a signature scheme, given any number of adaptively obtained message/signature pairs, it is computationally infeasible for an adversary to forge even a single new message/signature pair. The corresponding notion for digital signets is that, given any number of adaptively obtained valid signet pairs, it should be computationally infeasible for an adversary to generate a new valid signet pair, or indeed to generate any short string from which, with the aid of the extrication function and other public data, the content can be decrypted, without revealing information private to the adversary. We will discuss this connection – and why digital signatures do not solve our problem – more fully below.

The infeasibilty of generating valid signet pairs on demand is not sufficient to completely solve the problem, as there may be some alternative way to obtain the decryption key. In particular, the pirate might be able to find some short transmission to send to other users so that based on this short transmission the other users can easily compute the decryption key, without learning the sensitive information of the pirate.

We therefore desire an *incompressible* function f, defined as follows. Consider two communicating parties. One party (the adversary) possesses a short x (the signet pair) and wants to communicate to another party (a client) the long value $f(x)$, *without revealing* x. Roughly speaking, f is incompressible if no feasibly computable *short* message from the sender to the receiver simultaneously achieves both goals of enabling the receiver to compute $f(x)$ and hiding x.

For example, for a given prime p and generator g of Z_p^*, letting "$\circ$" denote concatenation, it seems plausible that the function

$$f(x) = g^x \circ g^{x^2} \circ g^{x^4} \circ \ldots \circ g^{x^{2^\ell}}$$

is incompressible under the Diffie-Hellman assumption. Incompressibility is defined more formally in Section 4. To our knowledge this appealing class of functions has not previously been studied.

Note that pseudo-randomness does not imply incompressibility. To see this, consider the original constructions due to Blum and Micali [4] and Yao [26]. In these constructions a one-way permutation g is repeatedly applied to a seed s, and in each iteration a hard-core bit is output. This is not incompressible because from $g(s)$ and the first bit of the sequence one can reconstruct the entire sequence, yet this information does not reveal s.

Our authorization schemes have a security threshold t: given up to t valid signet pairs, it is computationally infeasible to generate a *new* valid signet pair. Moreover, we conjecture that our scheme is actually *threshold t incompressible*, in the sense that given any t valid signet pairs (all mapping to the same key K under f) the adversary cannot find any short string to communicate K without revealing sensitive information belonging to the adversary.

The rest of this paper is organized as follows. Section 2 discusses the problem in more detail and states the *desiderata* for a solution. Section 3 discusses related work. Incompressible functions are defined in Section 4. Section 5 presents a basic solution and some optimizations. The basic solution and its variants require a special type of incompressible function called an *extrication function.* We suggest a candidate extrication function in Section 5. A possible attack on our function, and how to thwart the attack, are discussed in Section 6, along with other security considerations. The theoretical implications of the existence of incompressible functions are discussed in Section 7. Section 8 discusses application of digital signets to other distribution media.

2 Desiderata

As in the Introduction, we focus for concreteness on distribution via a CD-ROM. Let the CD-ROM contain an encryption of a large piece C of digital content. (The CD-ROM can in fact contain encryptions of many pieces of content, for example, several programs, or several images, or video clips, and our results apply to this case as well; for simplicity we restrict the discussion to a single large object C). The CD-ROM also contains in the clear some (small) programs, such as a program to extricate the key K from the signet pair, one for decrypting C given K, and possibly some others useful in communicating with the authorization center. We refer to these as *usage software*, to distinguish them from the encrypted content. We envision a scenario in which a user wishing to obtain C can telephone the authorization center, present a credit card number u, and obtain by dictation over the telephone a signet $\alpha = \alpha(u) = auth(A, z, u)$. Here, $auth$ is an authorization function, A is some privileged information known only to the authorization center, and z is part of the common information, available to the usage software. The user runs the extrication program on inputs (z, u, α), to obtain the decryption key K. This in turn is given as input to the decryption program, which can then unlock C.

This suggests our first few desiderata:

- The user's sensitive information u should be short and easy to communicate by telephone.
- The signet α should be short and easy to communicate by telephone.
- The decryption key K should be long and difficult to redistribute. Ideally, K should be as long as C itself.
- The extrication function should be (threshold t) incompressible, as described above (and defined more formally in Section 4).

One way of stretching a short signet pair to a long key K is by appealing to pseudo-randomness. We were not able to make this work using known pseudo-randomness technology. By definition of the problem, each user must be given a different signet, while a common extrication program must map all signet pairs to the same decryption key. However, the extrication function cannot simply combine the elements of the signet pair to obtain, say, a seed to a pseudo-random generator, since then the function would be compressible: a pirate could simply broadcast the seed. Using known constructions for pseudo-random generators and pseudo-random function generators we were not able to get around this problem.

A plausible approach to solving the illegal redistribution problem is to use digital signatures to authorize access to C. That is, the usage software could require as input the user's sensitive information together with the authorization center's signature on this information. The problem with this approach is that it is easy to patch the usage software: a sophisticated user can easily find the point in the code that tests for validity of the signature and modify the conditional "if valid then ..." to "if true then ...". Thus we also require, informally:

- The usage software should not be easily patched. In particular, there should be no conditional branch at which access is granted or denied.

If the extrication function is not (threshold t) incompressible, then we distinguish among several cases. If the adversary learns the secret information A, then it can actually behave as a pirate authorization center, generating valid signet pairs at will. This would be terrible, and we consider this a *total break*. Theoretically, the adversary could find an alternate means of generating valid pairs at will, without learning the secret information, which would still be disastrous and still qualifies in our view as a total break. In the threshold t scheme described in this paper these two capabilities are equivalent: the ability to generate additional valid signet pairs implies the ability to extract the secret information. Indeed in our scheme if the adversary can *existentially* break the scheme, generating even a single new valid signet pair, then it can totally break the scheme. In general (in contrast to the case with our scheme), an existential break does not imply a total break.

If the adversary is not able to perform an existential break but it can generate some other short string from which K can be reconstructed (without revealing its private information), we call this a *compression* break. We group existential and compression breaks together and call them *partial* breaks. If the adversary can only perform a partial break, then it would also have to distribute a patch – instructions on how to bypass the extrication function (for a compression break) or proper use of the extrication function (for an existential break). This might not be so bad (from the authorization point of view), since suspicious users might worry about importing a virus from what is obviously an unscrupulous user.

The authorization center must be able to quickly authorize a new user. It may use a trapdoor function to do this, or a function requiring some privileged information, or both. On the other hand, the extrication program runs on the user's machine, and can therefore be examined, say, using a debugger. Thus, we also require:

- Authorization can be computed quickly given the privileged information.

- Knowledge of the extrication function and up to t valid signet pairs should not reveal any useful information about the privileged authorization information.

3 Related Work

A first attempt at having a single decryption key K that is difficult for the users to share with impunity is described in a U.S. Pat. [20]. This patent describes a system in which each piece of content is encrypted with a different encryption key before it is distributed. A user wishing to purchase a piece of content obtains the encrypted content from a source (such as a widely distributed CD-ROM). The user then communicates a unique user-supplied customer identification number to an authorization center, which applies a secret "authorization function" to compute a *customer key*, unique to this customer. The customer key is used to encrypt K, the decryption key for the encrypted content. The customer is discouraged from sharing the authorization because it comes as a pair, the first component of which is the customer key which identifies the customer (to the digital content police).

The security of the system described in the patent [20] depends on the authorization function remaining secret to the end user. However, the authorization function can be completely determined by careful analysis of the calculation performed by the usage software to extricate the decryption key.

Early work on software protection systems focussed on physical modifications to the CPU [1, 19, 22]. Goldreich, while accepting the necessity of a hardware solution, extended the notion of protection to take into account what can be learned about a program by observing its input/output behavior as well as its sequences of memory access [16] (see also [23]). This approach does not apply to certain other types of digital content, such as audio or video, for which there is no concept of utilizing the content without being able to learn it.

The closest work in the literature is *Tracing Traitors* of Chor, Fiat, and Naor [8]. In that work, keys that allow decryption are distributed among users, and if any subset of fewer than t users collude and generate a new key, then at least one of the colluders will be *traced*. This works best in the broadcast encryption scenario [15], where the penalty for being traced is to be removed from the privileged set. There were no provisions in the tracing traitors work to tie a secret personal value (such as the credit card number) to the set of keys so that anyone who gets the illegally generated key can extract this information. However, the scheme in [8] can be modified to permit something similar. We compare the complexities of tracing traitors and signets in Section 5.6.

The closest work in the patent literature appears in a patent assigned to RSA Data Security Inc. [3]. We will discuss this patent in more detail after presentation of our schemes.

A different (but overlapping) approach, called *marking*, has been used for audio, video, and documents, although it is inappropriate for non-noisy data such as software [5, 6, 10, 12]. Marking solutions generally have two forms, both of which have the drawbacks that they rely on a digital content police and on non-automatic generation of suspicion. In one form, digital content, such as a photograph, is marked with some information secret to the creator, sometimes called a (hidden) *watermark*. This information should be cryptographically timestamped and saved by the creator. Upon finding an unauthorized copy of the material, the creator can produce strong evidence of ownership by analyzing a scan of the image and extracting the watermark. Different proposed schemes of this type have different attributes.

When the watermark is introduced in the spatial domain the processing is easy but the mark is hard to recover if the image has been re-scaled or warped (but easy if the image has simply been cropped) [12]. When the watermark is introduced in the frequency domain the computation is more intensive but recovery under distortion is easier [10]. In an extension of this idea, it is possible to personalize the watermarks to each legitimate recipient of the material, with the goal of tracing the source of any illegitimately re-distributed copies.

Finally, we note that portions of the work in this paper are covered by patent application [13].

4 Incompressible Functions

In the following definitions we let A and B represent two computationally bounded communicating parties. Consider a scenario in which A has an input x and there is a publicly computable function f. Intuitively, the goal is for A to communicate $f(x)$ to B *without revealing* x. In particular, we are interested in the case in which $f(x)$ is much longer than x.

A length-increasing function f is *incompressible* if, in order to communicate $f(x)$ to B in $o(|f(x)|)$ bits, A must reveal x, in the sense that B can effectively compute x. In other words, there is no feasibly computable short message that meets the goal of allowing B to learn $f(x)$ while simultaneously protecting x.

A stronger, and more complicated, definition can be made in analogy to the definitions for secure function evaluation, in which B learns no more about x than it can from $f(x)$ alone.

A *keyed* incompressible function $f(z,x)$ is a function such that once z is fixed, the function $f_z(x)$ is incompressible. Note that z can be as long as the output.

The crucial insight in developing a scheme for digital signets is that a mass distribution medium such as a CD-ROM contains enough space to hold a huge "hint" (the key z), that can be used in conjunction with the signet pair to compute the decryption key K. This motivates the remaining definitions.

A *publicly keyed* incompressible function $f(z,x)$ is a a keyed incompressible function that remains incompressible even when the key z is known (*i.e.*, is "public").

An *extrication* function is a publicly keyed incompressible function $f(z,x)$ for which there exists a corresponding polynomial time computable *authorization function auth* taking as input a set of (secret) values A, the public key z, and an identifier u such that, letting x_u denote the *valid pair* $(u, auth(A,z,u))$:

1. **Extrication** :$\forall z, u, u' : f(z, x_u) = f(z, x_{u'})$, that is, all valid pairs map to the same value;
2. **Incompressibility** :$\forall z : f_z(x)$ is incompressible.

More precisely, we need a *signet scheme generator* that, on input 1^n, produces in polynomial time $A, z, auth, f$ such that $auth(A,z,\cdot)$ and $f_z(\cdot)$ satisfy the extrication and incompressibility conditions for extrication functions (n is a security parameter).

A *threshold t* extrication function $f(z,x)$ is an extrication function such that for all

$$T = \{(u_1, auth(A,z,u_1)), \ldots, (u_t, auth(A,z,u_t))\},$$

$f_z(x)$ remains incompressible despite knowledge of T.

Strictly speaking, this too should be defined in terms of a generator. Accordingly, we define a threshold t signet scheme generator analogously to the definition of a signet scheme generator.

The next observation argues that any threshold t extrication function yields a scheme for digital signets. In terms of signets, z is part of the usage software or the common data; x_u is the signet pair for sensitive information (*e.g.*, credit card number) u, and A is privileged information known only to the authorization center.

Observation 4.1 *Any threshold t extrication function f that is sufficiently length-increasing on sufficiently short strings meets the specifications listed in Section 2 for digital signets.*

To see why Observation 4.1 holds, assume we have a threshold t extrication function f and its corresponding authorization function $auth$. Recall that when f is applied to a signet pair it produces a decryption key K. Let us paraphrase the list of specifications in the order in which they appear in Section 2.

1. The user's sensitive information u should be short.
2. The signet should be short.
3. The key K should be long.
4. The extrication function should be (threshold t) incompressible.
5. The usage software should not be easily patched. In particular, there should be no conditional branch at which access is granted or denied.
6. Authorization can be computed quickly given the privileged information.
7. Knowledge of the extrication function should not reveal any useful information about the privileged authorization information.

In light of the definitions in Section 4 some of these requirements are redundant.

The first three requirements are satisfied by the fact that f is sufficiently length increasing on sufficiently short strings. All the other requirements except the one about patching are satisfied by definition of a threshold t extrication function.

The requirement that there be no simple patch is informal. Informally, then, there is no simple patch because there is no decision point in the usage software. A key K' is obtained, which may or may not equal the decryption K. If the extrication function is given a valid signet pair then $K' = K$. As discussed above, threshold t incompressibility of $f_z(x)$ implies that without access to a, given t valid signet pairs it is still computationally infeasible to find any *new* short y from which k can be computed without revealing some private information.

5 Digital Signet Schemes

By Observation 4.1, in order to describe a digital signet scheme we need only describe an authorization/extrication pair of functions. Correctness of all our solutions rests on the existence of a group G in which discrete logs are hard and the Diffie-Hellman assumption holds [11]. (The Diffie-Hellman assumption is necessary for incompressibility, but not for security against a total break.) For example, G could be an elliptic curve; alternatively, let p be a prime such that $p-1$ has a prime factor $q > |U|$, and let G be a subgroup of Z_p^* of order q. Here U is a set of strings representing the set of potential users.

5.1 Outline of Construction

The construction is modular. In particular, the decryption key K is a sequence of blocks. We first define a *fundamental block* which takes a valid signet pair and produces a length n block of K (n is the security parameter). To create a key of length ℓn, where ℓ is a *stretch* parameter, we re-use the signet an additional $\ell - 1$ times (using different parts of the usage data z each time). Thus we create ℓ basic blocks of length n and concatenate the results to obtain K.

5.2 The Basic Block

Let g_1 be a random generator of G. Let $a, b_1, \ldots, b_t$ be secret, random, known only to the authorization center. Think of these as coefficients of a polynomial Q of degree t with free term a. Let $h_{1j} \stackrel{\text{def}}{=} (g_1)^{b_j}$, $1 \le j \le t$. We set $z \stackrel{\text{def}}{=} (g_1, h_{11}, \ldots, h_{1t})$. Intuitively, by the assumed hardness of the discrete log problem, z contains encodings of the coefficients $b_1, \ldots, b_t$ (but *not* of a), together with the generator. Define

$$auth^{(1)}(A, z, u) \stackrel{\text{def}}{=} (a - \sum_{j=1}^{t} b_j u^j) \bmod |G|,$$

where the superscript "(1)" is a reminder that we are describing the first of ℓ blocks in the general construction. Thus, the authorization function yields, on input u, $Q(u)$, the value of the secret polynomial at the point u. The extrication function is given by

$$f_z^{(1)}((u, \alpha)) \stackrel{\text{def}}{=} g_1^{\alpha} \prod_{j=1}^{t} h_{1j}^{u^j} = g_1^{\alpha} \prod_{j=1}^{t} (g_1^{b_j})^{u^j},$$

Note that this is just

$$g_1^{a - \sum_{j=1}^{t} b_j u^j} g_1^{\sum_{j=1}^{t} b_j u^j} = g_1^a.$$

(A similar approach in a different context is taken in [14, 24].)

5.3 Full Construction

Let $g_1, g_2, \ldots, g_\ell$ be random generators of G. Let $a, b_1, \ldots, b_t$ be secret, random, known only to the authorization center. Define $h_{ij} = (g_i)^{b_j}$, $1 \le i \le \ell$, $1 \le j \le t$, and let $z \stackrel{\text{def}}{=} (g_1, \ldots, g_\ell, h_{11}, \ldots, h_{\ell t})$. The authorization function remains unchanged. The extrication function is now the concatenation of ℓ applications of the extrication function $f_z^{(1)}$ described above, each using a different part of the public information z:

$$auth(A, z, u) \stackrel{\text{def}}{=} (a - \sum_{j=1}^{t} b_j u^j) \bmod |G|$$

$$f_z((u, \alpha)) \stackrel{\text{def}}{=} g_1^{\alpha} \prod_{j=1}^{t} h_{1j}^{u^j} \circ \ldots \circ g_\ell^{\alpha} \prod_{j=1}^{t} h_{\ell j}^{u^j},$$

where "$\circ$" denotes concatenation.

Consider the information known to the adversary that has corrupted t users, without loss of generality having sensitive information $u_1, \ldots, u_t$. Note that

$$\begin{pmatrix} u_1 & u_1^2 & \ldots & u_1^t \\ u_2 & u_2^2 & \ldots & u_2^t \\ \vdots & \vdots & & \vdots \\ u_t & u_t^2 & \ldots & u_t^t \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_t \end{pmatrix} = \begin{pmatrix} a - \alpha_1 \\ a - \alpha_2 \\ \vdots \\ a - \alpha_t \end{pmatrix}$$

where $\alpha_k = a - \sum_{j=1}^{m} b_j \cdot u_k^j \bmod |G|$, $1 \le k \le t$. Let us write this as $\mathcal{M} \cdot \vec{b} = \vec{a} - \vec{\alpha}$. In Section 5.5 we show that, since $\mathcal{M}$ is of full rank, we can translate any algorithm for totally breaking the scheme into an algorithm for extracting discrete logs. In Section 5.4 we describe a computationally more efficient solution, in which each α_k is a function of only a constant number of b_j's, independent of t. The corresponding equation involves a modified matrix $\mathcal{M}' \neq \mathcal{M}$. We will show that $\mathcal{M}'$ is also of full rank, and therefore the reduction to the security of discrete logs holds.

A static adversary chooses which parties to corrupt before execution of the protocol begins, and cannot alter its selection as the execution progresses. For the basic solution we prove perfect security against a total break by a static adversary. For the optimized solution we prove that for any set T of at most t parties chosen by the static adversary, with high probability the members of T working together cannot carry out a total break. The probability space is over choices made by the authorization center while preparing the individual signets.

5.4 Reducing the Computation Costs

In this section we remove the dependence on the threshold t from the computation costs, although we increase the storage requirements somewhat. The approach uses the expansion properties of random graphs and an adaptation of an idea of Siegel [25] for achieving t-wise (stochastic) independence.

Let $m = 2t$. As in the previous section, define $b_1, \ldots, b_m$ to be random, for $1 \le i \le \ell$ and $1 \le j \le m$ define $h_{ij} = (g_i)^{b_j}$, and let

$$z = g_1, \ldots, g_\ell, h_{11}, \ldots, h_{\ell m}.$$

Let $P : U \longrightarrow \{1, \ldots, m\}^3$ be a pseudo-random function that, on input $u \in U$, chooses an ordered 3-element subset $\{\beta, \gamma, \delta\}$ from $\{1, 2, \ldots, m\}$. Our first (but not final) idea is to set $\alpha = a - b_\beta u - b_\gamma u^2 - b_\delta u^3$ and let the signet be the pair $auth(A, z, u) = (\alpha, P(u))$. The extrication function f takes as input the signet pair $(u, (\alpha, (s_1, s_2, s_3)))$ (where if the signet pair is valid then $(s_1, s_2, s_3) = P(u)$); we write it this way since the usage software cannot compute or verify $P(u)$). Using z and the signet pair, the extrication software outputs

$$K = g_1^{\alpha} \prod_{1 \le j \le 3} h_{1 s_j}^{u^j} \circ g_2^{\alpha} \prod_{1 \le j \le 3} h_{2 s_j}^{u^j} \circ \ldots \circ g_\ell^{\alpha} \prod_{1 \le j \le 3} h_{\ell s_j}^{u^j}.$$

Thus, as in the original construction, K is the concatenation of ℓ components. For ease of discussion, we focus on just the first component in K for the remainder of this section.

The *intuition* in the optimization is as follows. Let us construct the bipartite graph $G = (U, M, E)$, where U is a set of vertices corresponding the the user credit card numbers u, M is a set of vertices corresponding to the b_j's, where by definition $h_{ij} = (g_i)^{b_j}$, and the edge $e = (u, j)$ is in E if and only if $j \in P(u)$. The following claim is proved in the full version of the paper. The claim is important since standard arguments do not yield the desired expansion for m as small as $2t$; on the other hand to minimize space requirements we need m to be as small as possible.

Claim 5.1 *Let T be a randomly chosen subset of U of size at most t. If P is a truly random function, then with probability at least $1 - 1/t^3$ (over choices made by P), for every $R \subseteq T$ $|\Gamma(R)| > |R|$ where $\Gamma(R)$ denotes the neighborhood of R.*

Again, speaking intuitively, this means that for any coalition of $r \leq t$ users, chosen adaptively given only that part of the graph induced by the vertices corresponding to the users already chosen, with high probability the information given to these users yields just r equations in at least $r+1$ unknowns (the unknowns are the coefficients b_j and a), and therefore should yield no information about the free term. However, the situation is a bit more complicated, as the equations may not be linearly independent.

Let T be a set of at most t users chosen by a static adversary. By Claim 5.1 the subgraph of G induced by T is expanding with high probability. Assuming we are in this high probability case, we will ensure independence as follows. Let $s = |P(u)|$. (We have taken $s = 3$ but the construction is general.) Replace each coefficient b_j by s coefficients $b_{j1}, \ldots, b_{js}$ chosen at random with the same distribution as b_j was chosen. Set

$$d := a - \sum_{j \in P(u)} \sum_{k=1}^{s} b_{jk} u^k.$$

In Lemma 5.1 we prove linear independence of any t equations of this form with distinct values of u. In Theorem 5.1 we show that independence implies security against a total break.

Consider the following $|U| \times s|M|$ matrix $\mathcal{M}$. $\mathcal{M}$ is most easily described as a $|U| \times |M|$ matrix of s-tuples. When we are discussing it in this form we will call it the *compressed* matrix. When we need to look at each element of the s-tuple individually, so that we are really looking at matrix of $s|M|$ columns, we will refer to it as the *expanded* matrix.

We denote an entry of the compressed $\mathcal{M}$ by $\mathcal{M}[uj]$, where $1 \leq u \leq |U|$ and $1 \leq j \leq |M|$. We denote an entry of the expanded $\mathcal{M}$ by $\mathcal{M}[uj]_x$, where $1 \leq u \leq |U|$, $1 \leq j \leq |M|$, and $1 \leq x \leq s$. Fix a value u. Let $P(u) = u_1, \ldots, u_s$, where $u_1 < u_2 < \ldots < u_s$. Then for $j \in P(u)$ let $\mathcal{M}[uj]$ be the s-tuple $(u, u^2, \ldots, u^s)$. All other entries in row u of the compressed matrix are s-tuples of zeros.

Lemma 5.1 *Let*

$$G = G_{U,M,P} = (U, M, \{(u, j) | u \in U \wedge j \in P(u)\})$$

Assume G has the property that, for a random set $T \subset U$ of cardinality at most t, with high probability the subgraph induced by T is an expander, in the sense that for all $R \subset T$, $|\Gamma(R)| \geq |R|$. Then for every random subset T of up to t rows of the expanded $\mathcal{M}$, with high probability the rows in T are linearly independent.

Proof: We show that if T, $|T| \leq t$, is a subset of U whose induced subgraph is expanding, then the corresponding set of rows in $\mathcal{M}$ is linearly independent. With a slight abuse of notation we identify rows of (compressed or expanded) $\mathcal{M}$ with vertices in vertex set U of $G = G_{U,M,P}$; similarly, we identify columns of the compressed $\mathcal{M}$ with vertices in vertex set M of G. Thus, we let T also denote the subset of vertices in vertex set U of G corresponding to the rows in T.

Suppose for the sake of contradiction that the lemma is false, and let T be set of rows forming a *minimal* counterexample. Thus, $|T| \leq t$, for all subsets $T' \subset T$, the rows of T' are linearly independent, and the subgraph of G induced by be vertices corresponding to the rows of T is expanding (as described in the statement of the lemma). Let $\mathcal{M}_T$ be the submatrix of $\mathcal{M}$ whose rows correspond to elements of T. Then there exists a row vector $\vec{x}$ such that $\vec{0} = \vec{x}\mathcal{M}_T$. The following claim is immediate from the minimality of T.

Claim 5.2 *No entry in $\vec{x}$ can be zero.*

Let us take any $s \times 1$ submatrix of column α of the compressed $\mathcal{M}_{T'}$ and expand it. We obtain an $s \times s$ Vandermonde matrix.

Since the Vandermonde matrix is of rank s, no linear combination of its rows yields $\vec{0}$. This yields a contradiction to the fact that, using the superscript "*" to denote the expanded version, $\vec{x}\mathcal{M}_T^* = \vec{0}$.

It remains only to prove the the following claim.

Claim 5.3 *Consider the expanded $\mathcal{M}_T$. If $\vec{0}$ can be expressed as a linear combination of the rows in $\mathcal{M}_T$, then some column of the compressed $\mathcal{M}_T$ contains at least one and at most s non-zero entries.*

Proof: To prove the claim it is sufficient to prove that, under the stated assumption, some vertex in $\Gamma(T)$ has at most s neighbors in T. To do this we will use the expansion property of the graph G.

For every $u \in U$, $|\Gamma(u)| = s$ (every element in the left side of the bipartite graph has exactly s neighbors). Thus $|\Gamma(T)| \leq s|T|$. But by the expansion property, $|\Gamma(T)| \geq |T|$. Thus, some element $v \in \Gamma(T)$ has at most s neighbors in T. ■

This concludes the proof of Lemma 5.1. ■

5.5 Security Against a Total Break

We now show that no set of t (or fewer) users can *totally* break the scheme, as long the matrix associated with their sensitive information $u_1, u_2, \ldots, u_t$ is of rank t. For simplicity we state the theorem and the proof only for the case where the group is a sub-group of size q of Z_p^* where both q and p are prime. However, it does generalize to other groups. The approach is to show that, given g^a, the adversary's view can be perfectly simulated. This is done along the lines of [14, 24].

Theorem 5.1 *If an adversary $\mathcal{A}$ controlling a set of users $u_1, u_2, \ldots, u_t$ can totally break the scheme with probability ρ (given that the associated matrix is of rank t), then we can find discrete logarithms in the group G with probability ρ in time similar to $\mathcal{A}$'s run-time.*

Proof: Recall that the key is a sequence of blocks. We assume that the scheme works as follows: for each u_k there are $u_{k1}, u_{k2}, \ldots u_{km}$ associated with u_k. We assume that the matrix

$$\mathcal{M} = \begin{pmatrix} u_{11} & u_{12} & \ldots & u_{1m} \\ u_{21} & u_{22} & \ldots & u_{2m} \\ & & \vdots & \\ u_{t1} & u_{t2} & \ldots & u_{tm} \end{pmatrix}$$

has rank t. There is a generator g and the first block of the key is g^a where a is secret and random. Random values $\vec{b} = b_1, b_2, \ldots b_m$ are chosen in $\{0, \ldots, q-1\}$. The signet of u_k is $\alpha_k = a - \sum_{j=1}^m b_j \cdot u_{kj} \bmod q$. Let $\vec{\alpha}$ be the list of α_k's given to the set of users controlled by the adversary. For $1 \le i \le \ell$, choose random generator $g_i = g^{c_i}$. The ith block of the key is g_i^a and let $h_{ij} = g_i^{b_j}$. As indicated above, for each u_k we have that $g_i^a = g_i^{\alpha_k} \prod_{j=1}^m h_{ij}^{u_{kj}}$.

First note that a total breaking implies that the adversary knows a (in the sense that it can be easily extracted). Therefore, we show how given g^a the above view may be simulated (perfectly), and hence the probability of success of $\mathcal{A}$ is the probability that we compute a. We may assume that $t = m$, since any matrix of rank t may be completed for a matrix of full rank. Note that the relationship between the α_k's and the b_k's is $\mathcal{M} \cdot \vec{b} = a - \vec{\alpha}$ which implies that $\vec{b} = \mathcal{M}^{-1}(a - \vec{\alpha})$. Thus, if $\mathcal{M}_{kj}$ denotes the k,j minor of U, then for $1 \le j \le t$

$$b_j = \sum_{k=1}^{t} (-1)^{j+k} \frac{|\mathcal{M}_{kj}|}{|\mathcal{M}|} \cdot (a - \alpha_k).$$

The simulation is therefore as follows

- Choose $\alpha_1, \alpha_2, \ldots \alpha_t$ at random.
- Set
$$h_j = g^{b_j} = \frac{(g^a)^{\sum_{k=1}^t (-1)^{k+j} \frac{|\mathcal{M}_{kj}|}{|\mathcal{M}|}}}{g^{\sum_{k=1}^t (-1)^{k+j} \frac{|\mathcal{M}_{kj}|}{|\mathcal{M}|} \alpha_k}}.$$
The expression for h_j implies efficient computation, since we can extract $|\mathcal{M}|$th roots efficiently over G.
- Choose random $c_1, c_2, \ldots c_\ell$ relatively prime to the order of the group.
- Set $g_i = g^{c_i}$ and $h_{ij} = h_j^{c_i}$.

It is not hard to show that the distribution of the h_{ij}'s, g_i's, α_k's and g^a is as in a real execution. ■

As mentioned, the proof can be extended to other groups: the principal difficulty is in taking $|\mathcal{M}|$th roots. The idea is simply to change the generator g into $g' = g^{|\mathcal{M}|}$. With g' we can effectively take the $|\mathcal{M}|$th root we need. This works in cases like elliptic curves and RSA.

5.6 Complexity of the scheme

The complexity of the scheme is composed of three components:

1. the storage requirements, or, more precisely, the ratio between the length of the key and the information used to encode it;
2. the size of the signet; this should be very small so as to allow transmission over the telephone through a human;
3. the computation required to compute the key given the signet and the stored information.

We require $t|K|$ bits to encode a decryption key K are $|K| \cdot t$ bits in a t-resilient fashion. The size of the signet is simply the size of an element in the group. This can be as small as 160 bits: we choose as our group the subgroup of size q of Z_p^*, where q and p are primes such that $q|(p-1)$ and q is less than 2^{160} (no good algorithms are know for discrete log in this case). The computation costs are quite high: the scheme requires ℓt exponentiations where ℓ is roughly $|K|/|G|$ in the basic scheme and $O(\ell)$ exponentiations in the improved one.

It is instructive to compare the signet scheme of this paper to an adapted version of the *tracing traitors* scheme of Chor, Fiat, and Naor [8][1]. In the signet scheme, the user's identifying number u is connected to the transaction; in the traitor tracing scheme it is just some number assigned to the user that is connected to the transaction. In terms of storage requirements, an improved version (not described in [8]) of the tracing traitors scheme requires $(|K| t \log |U|)$ bits to encode a decryption key K. The length of the signet (i.e. the number of bits communicated to the user) in this case will be roughly $\log^2 |U|$ times the size of a private key in the underling encryption method. The computation costs will be better than in the signet scheme, since they involve only private key computation (roughly $|K| \log^2 |U|$ encryptions).

To summarize, it seems that the main advantages of our signet scheme are the short signets and the fact that the connection between the signet and the user's private information is so direct. The other advantage is the storage. However, in terms of computation the tracing traitors scheme may be superior, since it requires only private key computation which, in general, is faster than exponentiation.

5.7 Comparison with Arcade

In this section we describe the Arcade system of RSA Data Security Inc. [3]. The system assumes a *distribution* channel for the commonly accessible material (*e.g.,* laser discs, CD-ROM, diskettes, TV cables, radio broadcast) and a special *access number* channel (the example given in the teaching of the patent is "diskettes offered in retail stores or by [conventional] mail"). This latter channel is used for the distribution of *access numbers* – long strings used in decrypting the encrypted content. Each access number A_i must be as long as the content itself. The set of access numbers plays a role similar to the public key in an extrication function. However, in the preferred embodiment described in the patent, each access number A_i can be used at most once (by at most one user), so this set is enormous. In contrast, in our system a *unique* public key z is used by *all* users – that is why we say that z is part of the common public data.

In the Arcade system, when a user wishes to obtain access to content C_j, she obtains an access number A_i from the access number channel, together with the index i, and she transmits the pair i, j to the authorization center. If A_i has not previously been used, the center responds with a short message t_{ij}. The access software takes as input A_i, t_{ij}, and

[1] Our terminology differs from that in [8], and what is referred to as storage there corresponds to signet size in our case.

the encryption of content C_j, and produces the cleartext of C_j. The principle difficulty is the distribution of the (long) access numbers.

The preferred embodiment of the Arcade system resembles the special case $t = 1$ in our original construction. Tolerance against coalitions is obtained by using each A_i only once. We briefly describe the details for a single piece C of content. Let $G = Z_p^*$ where p is a large prime (as long as the content) and let prime q divide $p - 1$. Let g generate a q-sized subgroup of G; q is short, say of length less than 200 bits. For random $0 \leq r_i, s \leq q - 1$, define $A_i = g^{r_i} \bmod p$ and $B = g^s \bmod p$. B is used as a one-time pad to encode the content. Thus the ciphertext is $C = B \oplus M$, where M is the plaintext.

Assume a user u obtains the pair (i, A_i) from the access number channel. Upon receipt of i from u, the authorization center checks that A_i has not previously been used. Assuming A_i is unused, the authorization center sends $t_i = s - r_i \bmod q$ to u. The extricaton software computes $g^{t_i} A_i = g^{s-r_i} g^{r_i} = g^s = B$. ¿From the one-time pad B it is then easy to retrieve M.

In terms of our scheme, the A_i provide a supply of coefficients which are never re-used. However, their distribution is essentially point-to-point; for example, it cannot take advantage of the cost-effectiveness of mass production.

6 Additional Security Considerations

In this section we discuss one potential compression attack, how to thwart it, and a general technique for minimizing damage in the case of a total break.

6.1 Compression Attack

The following attack against the full construction, described in Section 5.3 was suggested by Cem Celebiler [7]. Suppose that users $u_1, u_2, \ldots u_k$, $2 \leq k \leq t$, collaborate as follows. They will create a "fake identity" whose description is k group elements, together with a corresponding signet. (By the security against a total break, the traitors cannot obtain a new valid signet pair, so the fake identity cannot simply be an element of U) The coalition chooses random $c_1, c_2, \ldots, c_k$ summing to 1 mod $|G|$ (note that the adversary needs knowledge of $|G|$ to do this). Clearly

$$(g^a)^{c_1} \cdot (g^a)^{c_2} \cdot \ldots (g^a)^{c_k} = g^a.$$

Given k valid signet pairs $(u_1, \alpha_1), \ldots, (u_k, \alpha_k)$, the "fake identity" is $v_1, \ldots, v_t$, where $v_j = \sum_{i=1}^{k} c_i u_i^j$, and the corresponding signet is $\alpha = \sum_{i=1}^{k} c_i \alpha_i$. We have

$$g_i^a = g_i^\alpha \prod_{j=1}^{t} h_{ij}^{v_j}.$$

This completes the description of the attack. Note that $(v_1, \ldots, v_k, \alpha)$ is longer than a valid signet pair but shorter than the decryption key K.

There are several ways to thwart this attack. First, as noted above, the attack requires the adversary to know $|G|$ and so it is not applicable in the case of RSA where obtaining a non-trivial set $c_1, \ldots c_k$ summing to 1 mod $|G|$ is equivalent to factoring.

Consider next applying this attack to the computationally efficient scheme described in Section 5.4. Here the attack fails with high probability even when $|G|$ is known: the sensitive information of at least one of the traitors is apparent from the "fake identity." Specifically, when we construct the (compressed) matrix of sensitive information, with high probability there will be a column with a unique non-zero entry. This will yield a pair equations $cu = v_1$ and $cu^2 = v_2$. Dividing the second by the first modulo $|G|$ yields u.

Finally, consider the following modification of the full construction: use polynomials of degree $2t$ instead of polynomials of degree t. Now given $v_1, \ldots, v_{2t}$ we obtain $2t$ equations in $2k$ unknowns (recall, $k \leq t$), where the unknowns are $c_1, \ldots, c_k$ and $u_1, \ldots, u_k$. This is not entirely satisfactory, since the equations are not linear and may not be efficiently solvable (although they are of special form).

To summarize, this attack fails partially against the inefficient scheme and completely against the efficient scheme. This does not guarantee anything about security against other potential compression breaks. Indeed, we raise the possibility that there do not exist *any* incompressible schemes. We connect this question to communication complexity in Section 7.

Remark 6.1 *The same attack applies to the preferred embodiment of the Arcade system, as nothing in the design of that system precludes many users from obtaining the same A_i's. Ensuring that each A_i is used only once by the authorization center is not protection against the attack.*

6.2 General Technique for Minimizing Damage

The following general technique can minimize the damage caused by an adversary that succeeds in a total break.

Once the authorization center has calculated the signet x_u, it does not send it directly but instead sends what we call a *signet precursor* $\xi^{-1}(x_u)$, where ξ is a trapdoor function that is part of the usage software. The usage software applies ξ to the signet precursor in order to obtain x_u.

As a specific example, the function ξ can be $\xi(x) = x^r \bmod M$, where M is the product of two large primes and r is relatively prime to $\phi(M)$. Choosing r to be small (for example, 3) is useful since it reduces the amount of calculation performed by the usage software. The factorization of M is a secret known only to the authorization center, so the authorization center can compute $\xi^{-1}(x_u) = x_u^{1/r}$, while, under the RSA assumption, an adversary cannot. The signet precursor is now $\xi^{-1}(x_u)$.

Even if the adversary has observed enough legitimate authorizations to be able to calculate x_u, she cannot become an (undetectably pirate) authorizing center because she cannot calculate the signet precursor needed by the user's program. However, she can provide a patch that will allow a short key to be used instead of the long key K. This is true since M is small and the ξ calculation quick compared to storage and calculation of the rest of the extrication function.

7 On the Existence of Incompressible Functions

The security of the scheme described here against partial breaking relies on the the incompressibility of the function $f(x) = g_1^u \circ g_2^u \circ \ldots \circ g_\ell^u$, which we have not proved. Furthermore, we do not know whether incompressible functions exist at all, even under standard cryptographic assumptions.

Indeed, we show that the existence of incompressible functions implies a certain lower bound on communication length for secure function evaluation.

Suppose that Alice, having input x, and Bob, having input y, wish to compute $f(x, y)$ for some function f in a secure way, *i.e.*, so that each player does not learn more than can be efficiently deduced from its own argument and the value $f(x, y)$. Such protocols are known for any polynomial-time computable function [17, 21, 26]. However, the amount of communication in these protocols is proportional to the circuit size of f. We now argue that if there exist incompressible functions, then for certain functions f, secure function evaluation of f requires an amount of communication that is $\Omega(|f(x, y)|)$.

Let $f(u)$ be a presumed incompressible function, and consider a secure function evaluation protocol for the function $f(u, \cdot) = f(u)$. Using such a protocol the holder of u should be able to give away $f(u)$ without revealing too much information about u (not more than is revealed by the value of $f(u)$). If A (the holder of the u) and B engage in the secure protocol for f, then at the end of the protocol B knows $f(u)$ and nothing more. Furthermore, we can make the protocol run in a single round: since B has no input, its actions are determined by the message it receives and its internal coin flips. A can choose the random coin flips of B and simulate the messages in the protocol without B's help. Therefore we can think of B's coin flips as being fixed and A can send all the messages at once. If the secure function evaluation allows for a small probability of error, than we may not fix B's coin flips, but they can be chosen from a very small subset. A therefore chooses B's coins and adds them to the message. The incompressibility of f implies that any feasibly computable message that simultaneously protects the privacy of x and allows B to compute $f(x)$ is of length $\Omega(|f(x)|)$.

8 On Using Signets

We now discuss several issues related to the proper application of a signet scheme. The first question is, when should extrication be performed? One possibility is that it will be done only once, when the software is installed (or in general when the content is unlocked). This is indeed attractive, since it has the advantage that the computation of f does not interfere at all with the regular operation of the software, and we may anyway assume that the installation is a lengthy operation. Thus, adding even several minutes of CPU time to evaluate the extrication function f may be tolerable to the user. However, this assumes that all the parts protected by the key K are copied into the user's disk, which is not always the case.

Another possibility is to extricate the key when the software is being used. We observe that many classes of content have the property that they are accessed sequentially, or nearly so. This is certainly true of digital movies and music (at least within each music track). Even software will occasionally exhibit this type of pattern. For example, in a CD-ROM adventure game, the user moves from experience to experience in a way that is largely predictable. We can exploit these patterns in our scheme by segmenting the key such that only a small part of the key is used for a particular segment of the content. Since our long key is merely the concatenation of the results of logically separate calculations, this is a very natural step.

Two advantages accrue from this key segmentation. First, the latency seen by the user is reduced, as only a small part of the key need be calculated initially. The key(s) for the next segment(s) can be calculated on the fly as the user is experiencing the current segment. Second, the extrication program need not be designed with space for the entire key. Here the important point is not the space saved (although it can be significant), but rather the new hurdle this places in the way of the adversary. Remember, if the extrication function is incompressible, the adversary's only choice is to actually perform the calculation (and reveal his identifying information), or bypass the calculation by patching in the entire key. In the latter case he may find no convenient place to put it in the program. For example, in a processor with a segmented address space like a PC in 16-bit mode, it may be a nontrivial task to replace 16-bit offsets with 32-bit full addresses in every place needing the current key segment.

For concreteness we have been discussing CD-ROMs as the content media. However, signets are equally applicable to other media. In particular, a system whereby encrypted digital content is continually and repeatedly broadcast by satellite or cable, and customers purchase signets for the content that interests them, is appealing (this is the main scenario of broadcast encryption of [8, 15]). In fact, the current system for distributing broadcast and cable TV networks to rural American satellite dishes is what we would call a signet scheme in all respects except one. The authorization signal sent to legitimate decoders is the TV network's key K encrypted with the particular decoder's identification number, for each network to which the user has subscribed. However, the extrication function that such a scheme compressible, and in fact the attack used by the estimated 50% illegal decoders is to patch in the key K itself. There is an elaborate underground network to distribute these so-called "wizard keys" each month, when they are changed. Since they are the same number of bits as the legitimate authorization signals, the pirate distributors are not significantly disadvantaged.

Note the significance of the storage requirements in this scenario: the g_i's are not stored by the users' decrypting boxes ahead of time, but every so often, say once a minute, the center broadcasts the new g_i's and h_{ij}'s. Therefore, the longer their length, the larger the percentage of bandwidth that is devoted to this. Moreover, there is good reason to transmit the g_i's even more often than when they are changed: in general the users' boxes do not follow all the channels to which they subscribe, and when the user tunes in to a channel, the box needs to get the new key quickly.

Another applicable scenario, also mentioned in [8], is that of on-line services and distributed databases. Here the pirate cannot hope to redistribute all the decrypted content, since some of it may not yet have been generated. An instructive example is an on-line newspaper. A user receives a fixed signet upon subscription to the newspaper. New encryption keys K and corresponding g_i's and h_{ij}'s are chosen for each issue and distributed with the issue. Even if we set $t = 1000$ and choose a one thousand bit prime p, the amount of additional storage we need to provide the legitimate users in order to decrypt an issue is less than 200K bytes, on the order of the size of a picture! Moreover, any pirate trying to offer access to the system must guarantee that each time a new decryption key K is issued (daily, in the case of a newspaper), the pirate will distribute the new key. This increases the vulnerability of the pirate and changes the nature of the interaction with the pirate from one-time to ongoing.

We feel that the attractiveness of a signet scheme, and implicitly the value of incompressible functions, derives from the economic leverage between inexpensive broadcast means (and by that we include stamping out CD-ROMs) for distributing encrypted, not-yet-bought content, and more expensive point-to-point distribution of immediately usable content. Currently, magnetic disks cost two orders of magnitude per megabyte more than CD-ROMS, and recordable CDs (CD-R) are one order of magnitude more expensive (even discounting the more expensive recorder/player). This is significant leverage, but one might question whether it is just due to an accident in time–whether in the progression of technology this discrepancy will disappear. Indeed it may for physical media, but it has always been true, and seems inherent, that broadcast communication is less expensive per communicating party than point-to-point communication. Thus incompressible functions seem to be generally useful.

9 Directions for Future Research

As discussed, although we have proved security of our scheme against a total break, we have not proved incompressibility. The principal open problem suggested by this work is to prove or disprove the existence of incompressible functions based on standard cryptographic assumptions.

Our scheme relies on the incompressibility of the function $f(x) = g_1^x \cdot g_2^x \cdot \ldots \cdot g_\ell^x$ for $g_1, \ldots, g_\ell$ known. Prove equivalence of this assumption to security of our scheme against a compression break.

10 Acknowledgements

The authors are grateful to David Fuchs, whose tireless assistance with Mathcad helped guide the proof of Claim 5.1, and to Bernard Dwork for many discussions about linear algebra, the subject finally having piqued the interest of his daughter. Thanks also go to Noga Alon, Cem Celebiler, Amos Fiat, Joe Halpern, Joe Kilian, Michael Luby, and Larry Stockmeyer for helpful discussions.

References

[1] D. Albert and S. Morse, *Combating Software Piracy by Encryption and Key Management*, Computer, April 1984.

[2] *The Arcade Project: A Progress Report,* Ciphertext: The RSA Newsletter, Vol. 2, No. 1, 1994, p. 6.

[3] U.S. Patent 500,403, issued March 21, 1995, assigned to RSA Data Security, Inc., Redwood City, CA.

[4] M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Computing 13, pp. 850-863, 1984.

[5] D. Boneh and Shaw, *Collusion Secure Fingerprinting for Digital Data*, Proc. Advances in Cryptology - Crypto '95, LNCS 96, Springer, pp. 452–465, 1995. Full paper available: http://www.cs.princeton.edu/ dabo/publications.html.

[6] J. Brassil, S. Low, N.F. Maxemchuck, and L. O'Gorman, *Electronic Marking and Identification Techniques to Discourage Document Copying*, 13th Annual Joint Conference of the IEEE computer and Communication Societies, Proc. 1III Info-Comm'94, pp. 1278–1287.

[7] Cem Celebiler, *personal comm.*; cem@theory.lcs.mit.edu

[8] Chor B., A. Fiat and M. Naor, Tracing traitors, *Advances in Cryptology - CRYPTO' 94*, LNCS 839, Springer, pp. 257–270, 1994. (Full version available.)

[9] B. Chor and S. Goldwasser and S. Micali and B. Awerbuch, Verifiable secret sharing and achieving simultaneity in the presence of faults, FOCS 85, 1985, pp. 383–395.

[10] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, *Secure Frequency Domain Watermarking for Multimedia*, manuscript, 1995

[11] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE. Transactions on Information Theory, Nov. 1976, pp. 644–654.

[12] World Intellectual Property Organization publication PCT/US94/13366, May 1995.

[13] C. Dwork, J. Lotspiech, and J. Halpern, *Method and System for Protection of Digital Information* U.S. Patent Application (IBM Ref. No. AM9-95-026), filed 8/21/95

[14] P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, FOCS 28, 1987, pp.427–437.

[15] A. Fiat and N. Naor, *Broadcast Encryption*, Proc. Advances in Cryptology - Crypto '93, 1994, pp. 480–491.

[16] O. Goldreich, *Towards a Theory of Software Protection*, Advances in Cryptology – CRYPTO 1986, Springer-Verlag, 1987, pp.426–439.

[17] O. Goldreich, M. Micali, A. Wigderson, *How to play any mental game*, Proc. 19th ACM Symp. on Theory of Computing, 1987, pp. 218–229.

[18] Goldwasser, S. and S. Micali and R. Rivest *A secure digital signature scheme*, SIAM J. on Computing 17, 1988, pp. 281–308.

[19] A. Herzberg and S. Pinter, *Public Protection of Software* Advances in Cryptology – CRYPTO 1985, Springer-Verlag 1986, pp. 158–179.

[20] *US Patent No. 5,319,705 : Method and system for multimedia access control enablement*, Inventors: Halter, Bernard J., Le, An V., Co, Alphonse M., Johnson, Donald B., Matyas, Stephen M., Randall, James D., and Wilkins, John D., Issued to IBM Corp. 06/07/1994; Application Date:10/21/1992

[21] J. Kilian, **Use of Randomness in Algorithms and Protocols**, MIT Press, Cambridge, Massachusetts, 1990.

[22] S. Kent, *Protecting Externally Supplied Software in Small Computers,* Ph.D. thesis, MIT/LCS/TR-255, MIT, 1980.

[23] R. Ostrovsky, *Efficient Computation on Oblivioius RAMs*, STOC 22nd, pp. 514–523, 1990.

[24] T.P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, Proc. Advances in Cryptology - Crypto '91, Lecture Notes in Computer Science No. 576, Springer, 1992, 129–140.

[25] A. Siegel, *On universal classes of fast high performance hash functions, their time-space tradeoff and their applications*, FOCS 1989, pp. 20–25.

[26] A.C. Yao, *How to Generate and Exchange Secrets*, Proc. of the 27th IEEE Symp. on Foundations of Computer Science, 1986, pp. 162–167.