Key Compression and its Application to Digital Fingerprinting

Josh Benaloh

Microsoft Research

Abstract. Digital fingerprinting technologies are becoming an increasingly important tool to protect valuable content and other intellectual property. This paper describes an efficient method whereby any watermarking technology can be utilized to construct digital fingerprints that can distinguish individual instantiations of protected data without requiring replication of the data. This technology enables large amounts of data to be selectively distributed to large numbers of people over a limited medium such as a broadcast channel or CD-ROM. An application of this technology – protection of motion pictures for in-flight entertainment systems – will be specifically discussed.

Note: This is a reprint of an article that was originally submitted for publication in April 2001.

1 Introduction

There are many instances in which one has a large database or other information set to which different entities are to be granted access to different subsets of the data. While databases are a natural example, another example might be pay-per-view broadcasts in which each customer has purchased viewing rights to a different set of programs. A third example is seen in fingerprinting methodologies wherein multiple copies of each "clip" of content are produced and each recipient is to be given access to exactly one of the copies of each clip.

It is natural in all of these scenarios to encrypt each datum separately and make the encrypted data set available to all subscribers. Each individual subscriber can then be given exactly those keys that protect the data to which that individual is entitled. Thus, the problem of distributing different data sets is reduced to the problem of distributing different key sets.

A problem, however, is that if the universe of keys is large, the subsets of keys which must be customized and separately sent to each individual may be large. This may impose substantial burdens — even if the number of subscribers is smaller than the number of keys.

This paper describes a mechanism whereby each key set can effectively be compressed into a single key whose size is, for all intents and purposes, dependent only on a security parameter. As an example, the number of distinct keys that can be compressed into a 1024-bit value is limited only by the number of distinct primes less than 2^{1024}. While recovery of an individual key requires time proportional to the security parameter and the number of keys compressed for that subscriber, the additional time to compute more than one key is small and independent of the number of keys compressed. Specifically, if n keys are compressed for a subscriber, any set of m of these compressed keys can be recovered in time $\mathcal{O}((n + m \log m)s)$, where s is a security parameter.

A primary application of this mechanism is in the domain of digital fingerprinting for content protection. Many small portions of a film, song, or other data stream can be selected by its owner. Two (or more) copies of each of these small "clips" can be created with minor differences in each copy. Each copy of each such clip can then be encrypted with a separate key. By giving distinct sets of keys to distinct users, one can *fingerprint* each instance of the content so that, if an instance is later detected being mis-used, the instance may be able to be traced back to its source.

Of course, cryptographic keys are generally incompressible and distinct keys should normally be chosen independently. However, in many circumstances, computational — rather than information theoretic — independence can suffice. A pseudo-random number-generator can certainly be used to expand a single seed (or key) into a virtually unlimited number of computationally independent keys. The challenge here, however, is to give different seeds to different subscribers so as to enable subscribers to recover exactly those keys to which they are entitled without allowing them to compute any of the remaining keys.

2 Related Work

The methods used for key compression in this paper are similar to techniques for key management developed in [ChTa89] and later used in [HaPe95]. However, some of the details are different and the application to digital fingerprinting is entirely new.

The approach of segmenting digital content and separately marking and encrypting multiple copies of each segment is discussed in [BoSh98]. The techniques described herein expand upon this work by enabling more efficient use of constrained distribution media.

The application to content distribution can, from a certain perspective, be seen as a dual to broadcast encryption ([FiNa93]). The methods of broadcast encryption can solve the basic problem described in the introduction by broadcasting each key to the set of subscribers eligible to receive that key.

The bandwidth required by broadcast encryption is proportional to the number of keys but (after initialization) independent of the number of subscribers while the computation required by each subscriber to recover each key is proportional to the number of subscribers. The methods that will be described herein require bandwidth that is proportional to the number of subscribers and independent (after initialization) of the number of keys while the computation required by each subscriber to recover each key is proportional to the number of keys (although amortization can reduce the cost of each additional key to the log of the number of keys recovered).

Thus, in an environment of constrained bandwidth, broadcast encryption works well if the number of subscribers is large relative to the number of keys while the methods herein are better when the number of keys is large in relation to the number of subscribers. The reverse is true if computation rather than bandwidth is the limiting factor. Both methods work equally well when the numbers of keys and subscribers are comparable.

Another advantage of the methods of this paper offer over broadcast encryption is that collusion is *not* a concern. Any set of colluding parties can obtain precisely the union of the sets of keys to which they are individually entitled. No advantage is obtained towards discovering any key to which none of the parties is entitled.

3 Preliminaries

We begin by giving a generic definition of a *key generation system.*

Definition 1. *Let m be a positive integer and let $T_1, T_2, \ldots, T_n$ be a collection of index sets with each $T_j \subseteq \{1, 2, \ldots, m\}$. We say that a collection of polynomial-time algorithms $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ is a* **key generation system** *if $\mathcal{A}(T_1, T_2, \ldots, T_n) = (K = \langle k_1, k_2, \ldots, k_m \rangle, V = \langle v_1, v_2, \ldots, v_n \rangle)$ and for each j such that $1 \leq j \leq n$, $\mathcal{A}_j(v_j) = \{k_i : i \in T_j\}$.*

Thus, the output of the *key generation function* $\mathcal{A}$ is a set of keys (one for each datum in a data set of size m) and a set of values each of which enables the derivation of a pre-defined subset of the keys.

The intent here is that the key generation function is given a set of mappings describing the data to which each subscriber is to be given access. Its output is a set of keys to be used to encrypt the data and a set of *compressed key sets* which can be distributed — one per subscriber — such that each subscriber can obtain those keys, and only those keys, used to encrypt data to which the subscriber is to be given access.

Note that this definition is trivially satisfied by the generation function that generates a random (or pseudo-random) set of keys $K = \{k_1, k_2, \ldots, k_m\}$ and individually doles out to each subscriber the entire set of keys $v_j = \{k_i : i \in T_j\}$ to which the subscriber is entitled. However the goal of this work is to develop a compression technique which allows the v_j to be much smaller than that which can be achieved by simple enumeration.

It should also be noted that the above definition can be trivially satisfied by broadcasting the entire key set K to all participants ($v_j = K$, for all j). Since our goal is to develop a key generation system in which each participant gets those keys *and only those keys* to which it is entitled, we must next define what it means for a key generation system to be *secure*.

Definition 2. *Let m be a positive integer and let $T_1, T_2, \ldots, T_n$ be a collection of index sets with each $T_j \subseteq \{1, 2, \ldots, m\}$. Let $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ be a key generation system. The system is said to be* ε**-secure** *if there exists no (probabilistic) polynomial-time algorithm that when given a partial list, $v_{j_1}, v_{j_2}, \ldots, v_{j_\ell}$ where $\{j_1, j_2, \ldots, j_\ell\} \subseteq \{1, 2, \ldots, n\}$, of the compressed key sets output by $\mathcal{A}$, outputs, with probability greater than ε, any $k_i \in K$ such that $i \notin \cup(T_{j_1}, T_{j_2}, \ldots, T_{j_\ell})$.*

In short, a key generation system is ε-secure if no polynomial-time algorithm can construct, with probability greater than ε, any key in a key set which is not derived from at least one of its inputs. Thus, no collusion of participants can gain access to any key to which none of the individual colluders have access. Note that in practice, the ε here is likely to be a function of a security parameter.

4 Key Compression Method

We assume that we begin with a large data set. We associate a small odd prime integer with each individual datum to be protected. No prime is associated with more than one datum, and the set of these primes $U = \{p_1, p_2, \ldots, p_m\}$ (and their association) is completely public. For each subscriber S_j, let $U_j \subseteq U$ be the set of primes associated with data to which S_j is entitled. That is, $U_j = \{p_i : i \in T_j\}$.

The owner (dealer) of the data selects two large prime integers Q_1 and Q_2 and forms their product $N = Q_1 Q_2$. Although it is not strictly required, it is preferred that Q_1 and Q_2 each be safe primes (one greater than twice a prime). What is specifically required is that no $p_i \in U$ divide either $Q_1 - 1$ or $Q_2 - 1$.

The owner of the data also selects a random value $x \in Z_N^*$ — the multiplicative subgroup of the integers modulo N. Of these values, only N is made public. For convenience, we also define the value $y = x^E \bmod N$ where $E = \prod_i p_i$ and this product is taken over all primes $p_i \in U$. There is no harm in making y public.

Next, we define $r_i = x^{E/p_i} \bmod N$. Thus, r_i is the (modulo N) p_ith root of y. The construction of N ensures that p_i does not divide $\varphi(N)$ and therefore that there is a unique p_ith root of every $y \in Z_n^*$. The datum associated with the prime p_i is encrypted (using a symmetric encryption algorithm) with the key k_i derived from r_i by a standard function such as $k_i = \text{SHA-1}(r_i)$ to remove the algebraic structure.

For each subscriber S_j, we define $E_j = \prod p_i$ where this product is now taken over all primes in U/U_j. In other words, E_j is the product of all primes associated with data to which subscriber S_j is *not* entitled. The owner of the data computes, for each subscriber S_j, the value $v_j = x^{E_j} \bmod N$. Each compressed key set v_j is then sent to subscriber S_j over a private channel (or encrypted for S_j and sent over a public channel).

It is now a simple matter for subscriber S_j to compute any key k_i for which it is entitled to the associated data. It is also clear (the formal proof will be given later) that subscriber S_j cannot compute any key k_i to which it is not entitled unless S_j has the ability to extract roots modulo N. A simple corollary is that there is no threat posed by collusion: if no members of a colluding set have access to a particular datum, then the collective information held by all of the members does not permit the computation of the associated key!

The computational independence of the keys is a direct consequence of Shamir's root independence lemma ([Sham83]) which shows that a set of (modulo N) p_ith roots of y is of no use in computing a (modulo N) rth root of y unless r divides $\prod p_i$. Thus, no collusion of subscribers that does not have access to a particular datum (say one associated with the prime p_i) will be able to jointly obtain access to that datum since they together have a set of roots of which none of their defining exponents are divisible by p_i.

5 Amortized Key Recovery

To recover any single key k_i, it is apparent that a subscriber needs to take its compressed key set v_j and raise it (modulo N) to the power of all primes, other than p_i, in the set U_j. Thus, the computational cost of obtaining any single key grows linearly with the number of data to which a subscriber is granted access.

However, if a subscriber wants to recover more than one key at a time, the amortized costs shrink rapidly. For example, a subscriber can compute two separate keys with only one small-prime modular exponentiation more than is required to compute a single key. This can easily be accomplished by raising the compressed key set v_j to the power of all primes in the set U_j other than the two distinguished primes that are associated with the two desired keys. This intermediate value can then be separately raised to the power of each of the two remaining primes to form the pre-image of each of the two desired keys.

In general, if m keys are to be recovered, a tree can be formed with a value at the root equal to the compressed key set v_j raised to the (modulo N) power of all primes in the set U_j other than those to be recovered. A balanced binary tree is now constructed by associating each of the remaining primes of U_j with a leaf. Each node of the tree will now contain the compressed key set v_j raised to the power of all primes in the set U_j other than those associated with the leaves of its sub-tree. In can readily be seen that m small-prime modular exponentiations must be done at each of the $\log m$ levels of the tree in order to complete the tree. The values at the leaves correspond to the m newly recovered keys.

6 Proofs of Properties

It is actually relatively easy to build upon past work to show that this scheme satisfies the desired properties.

Theorem 1. *The system of section 4 is a key generation system (according to the definition of section 3).*

Proof. Section 4 describes the procedures for reconstituting keys. Thus, it only remains to be shown that the key generation function and each of the key reconstruction functions operate in polynomial time.

The key generation function requires the time to generate and multiply two primes (quartic in the length of the primes) plus $\mathcal{O}(m+n)$ small prime modular exponentiations — where m is the number of keys and n is

the number of subsets. Since a modular exponentiation can be performed in cubic time, the key generation function is linear in the number of keys and subsets and quartic in the security parameter.

Each of the n key reconstruction functions requires $\mathcal{O}(m \log m)$ small prime modular exponentiations by the methods of section 5.

Thus, the system is polynomial-time as required. □

To show the security of the system, we must begin with a standard assumption.

Assumption 1 (RSA) *There is no polynomial Q and probabilistic polynomial-time algorithm $\mathcal{A}$ which takes parameters n, e, and x, (where n is a product of two equal length primes and e is relatively prime to $\varphi(n)$) and produces, with probability $\frac{1}{Q(|n|)}$, a y such that $x = y^e \bmod n$.*

Theorem 2. *If the RSA assumption is true, then for any polynomial ρ, the key generation system of section 4 is $\frac{1}{\rho(s)}$-secure — where s is the security parameter of the system.*

Proof. (sketch)

Let $K = \{k_1, k_2, \ldots, k_m\}$ be a collection of keys and let $T_1, T_2, \ldots, T_n$ with each $T_j \subseteq \{1, 2, \ldots, m\}$ be a collection of indices. Let $p_1, p_2, \ldots, p_n$ be a set of distinct odd primes of length at most s (the security parameter). We assume that m and n are bounded by a polynomial in s. Let $\mathcal{A}(K, T_1, T_2, \ldots, T_n)$ be the randomized key generation function defined as in section 4 which operates as follows:

1. select two "master" primes uniformly of length $\frac{s}{2}$ such that neither master prime is one greater than a multiple of any of the primes $p_1, p_2, \ldots, p_n$,
2. form the product N of the two master primes of the previous step,
3. uniformly select an integer x in the multiplicative group Z_N^*,
4. output N together with the vectors $K = \langle k_1, k_2, \ldots, k_m \rangle$ where each $k_i = x^{E/p_i} \bmod N$ and $E = \prod p_i$ and $V = \langle v_1, v_2, \ldots, v_n \rangle$, where each $v_j = x^{E_j} \bmod N$ and $E_j = \prod_{\{i : k_i \notin T_j\}} p_i$.

Suppose further that $J = \{j_1, j_2, \ldots, j_\ell\} \subseteq \{1, 2, \ldots, n\}$ and $\mathcal{B}$ is a polynomial-time algorithm which takes as input N and v_j for all $j \in J$ and outputs, with probability at least $\frac{1}{\rho(s)}$ (for some polynomial ρ), some $k_i \in K / \cup_{j \in J} T_j$.

Shamir's Root Independence Lemma ([Sham83]) asserts that this implies there must be a polynomial-time algorithm $\mathcal{B}'$ which takes as input only N and outputs, with probability at least $\frac{1}{\rho'(s)}$ (for some polynomial ρ'), some $k_i \in K / \cup_{j \in J} T_j$.

Since m and n must be polynomial in s, and since for any fixed set of primes $p_1, p_2, \ldots, p_n$, the special form master primes, as described in step 1 above, constitute a polynomial sized fraction of all the primes, this algorithm violates the RSA assumption. □

7 Digital Fingerprinting

An application of this form of key compression is to support a particular paradigm for fingerprinting of digital content. Fingerprinting is effected by providing, on a limited communication medium (such as a digital video disk or broadcast channel), two slightly different versions of several separate portions of the digital content. Each version of each portion is separately encrypted, and each content playback device is given the decryption keys for exactly one version of each portion. In this manner, any content which is leaked will contain a distinct pattern of versions showing which keys (and presumably which content player) this content originated on. Of course, in this context collusion presents a serious threat, since even when no set of players can obtain even a single key to which they are not entitled, they may be able to defeat the fingerprinting by mixing different content portions to which they *are* entitled. However, techniques such as those found in [BoSh98] can be introduced for collusion-detection.

The problem with employing this fingerprinting technique is that each content player must now receive essentially half of all of the decryption keys. The keys may actually be distributed (encrypted for each and every active player) on identical digital video disks together with the encrypted content. While there may be enough space on these disks to comfortably encrypt one key for each content player, if the number of keys each content player must receive grows large, space becomes a problem. Broadcast encryption offers a possible solution here, but the time to unpack each key is proportional to the number of active content players – which can be prohibitive. Instead, the method introduced in this paper can be used to effectively compress keys. This can reduce the space requirements to acceptable levels without requiring excessive time to unpack keys.

This precise fingerprinting and key compression paradigm is being incorporated into an in-flight entertainment standard currently being written by the World Airline Entertainment Association.

8 Alternate Formulations and Open Problems

It is possible to generalize somewhat the requirements for effective key set compression by utilizing any associative one-way hash function or one-

way accumulator ([BeMa93]). To do this, one begins by choosing a secret starting value x and associating a public value r_i with each datum D_i to be encrypted. The key to encrypt a given datum D_i is derived from the (associative) hash whose inputs are the secret starting value x and all of the public values r_j with $j \neq i$.

Each subscriber is given the (associative) hash whose inputs are the secret starting value x and all of the public values r_j for which the subscriber is *not* entitled to the corresponding datum D_j. It is now an easy matter for the subscriber to compute the key for any datum D_j to which to which the subscriber *is* entitled. Furthermore, the amortization tree method of section 5 works equally well here.

When a one-way accumulator is substituted for the associative hash function and the accumulator operation is exponentiation modulo a composite, then this is equivalent to the method of section 4.

Unfortunately, modular exponentiation is the only known method for constructing an associative one-way hash or accumulator. It would be useful to find other instantiations of these primitives. It would also be desirable to find weaker primitives which might suffice to achieve the results of this paper.

9 Conclusions

This paper describes a paradigm for effective compression of keys and its application to digital fingerprinting. Data and keys can be transmitted via a severely contrained channel so as to allow many subscribers to each obtain access to individually customized data sets. A primary application of this technique yields an extremely efficient mechanism for digital fingerprinting that offers substantive advantages over previous methods.

References

[BeMa93] **Benaloh, J.** and **de Mare, M.** "One-Way Accumulators: A Decentralized Alternative to Digital Signatures". *Advances in Cryptology — EuroCrypt 93*, ed. by T. Helleseth in Lecture Notes in Computer Science, vol. 765, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1994), 274–285.

[BoSh98] **Boneh, D.** and **Shaw, J.** "Collusion secure fingerprinting for digital data". *IEEE Transactions on Information Theory*, vol 44, no. 5 (1998), 1897–1905.

[ChTa89] **Chick, G.** and **Tavares, S.** "Flexible Access Control with Master Keys". *Advances in Cryptology — Crypto 89*, ed. by G. Brassard in Lecture Notes in Computer Science, vol. 435, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1990), 316–322.

[FiNa93] **Fiat, A.** and **Naor, M.** "Broadcast Encryption". *Advances in Cryptology — Crypto 93*, ed. by D. Stinson in Lecture Notes in Computer Science, vol. 773, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1994), 480–491.

[HaPe95] **Halevi, S.** and **Petrank, E.** "Storing Classified Files". *Unpublished manuscript*.

[Sham83] **Shamir, A.** "On the Generation of Cryptographically Strong Pseudo-random Sequences". *ACM Transactions on Computer Systems*, vol. 1, no. 1, ACM, New York (1983), 38–44.