PAPER *Special Section on Cryptography and Information Security*

The Sibling Intractable Function Family (SIFF): Notion, Construction and Applications

Yuliang ZHENG†, ***Nonmember***, **Thomas HARDJONO**††, ***Member*** ***and*** **Josef PIEPRZYK**†, ***Nonmember***

SUMMARY This paper presents a new concept in cryptography called the *sibling intractable function family* (SIFF) which has the property that given a set of initial strings colliding with one another, it is computationally infeasible to find another string that would collide with the initial strings. The various concepts behind SIFF are presented together with a construction of SIFF from any one-way function. Applications of SIFF to many practical problems are also discussed. These include the hierarchical access control problem which is a long-standing open problem induced by a paper of Akl and Taylor about ten years ago, the shared mail box problem, access control in distributed systems and the multiple message authentication problem.

key words: *information, security, cryptography*

1. Introduction

This paper presents a new concept in cryptography called the *sibling intractable function family* (SIFF). SIFF is a generalization of the concept of the universal one-way hash function family introduced in Ref.(13), and it has the property that given a *set* of initial strings colliding with one another, it is computationally infeasible to find another string that would collide with the initial strings. We also present a simple method for transforming any universal one-way hash function family into a SIFF. As Rompel has proved that universal one-way hash function family can be constructed from any one-way function,[16] we obtain the theoretically optimal result that SIFF also can be constructed from any one-way function.

SIFF has many nice features, and can be applied to a number of cryptographic problems. We will describe in detail a solution to the hierarchical access control problem, which includes a way to generate and update keys for a hierarchical organization. In a hierarchical organization it is assumed that authority is arranged in a hierarchical manner, where higher level members of the organization have access to resources and data classified at a lower level. In this way we solve, under the weakest assumption of the existence of any one-way function, a long-standing open problem induced by a paper of Akl and Taylor about ten years ago.[1] Applications of SIFF to other three problems, namely the shared mail box problem, access control in distributed systems and the multiple message authentication problem, will also be discussed.

The remainder of the paper is organized as follows. In Sect. 2, we give basic definitions for one-way functions, pseudo-random function families and universal hash function families. In the same section we also introduce the new notion of SIFF. In Sect. 3, we show a method for transforming any universal one-way hash function family into a SIFF. As a corollary, we obtain the theoretically optimal result that SIFF can be constructed from any one-way function. In Sect. 4, we give a formal definition for the security of a key generation scheme for hierarchical organizations, and present a solution to the hierarchical access control problem by the use of SIFF and pseudo-random function families. We describe in detail the following three aspects of the solution: key generation, key updating and proof of security. In Sect. 5, we suggest three other applications of SIFF to show its usefulness. The first application is to the shared mail box problem, the second to the access control in distributed systems and the third to the multiple message authentication problem. Section 6 closes the paper with a summary of the results and a suggestion for further research.

2. Basic Definitions

In this section we introduce the definitions for one-way functions, pseudo-random function families, universal hash function families and sibling intractable function families.

2.1 Pseudo-Random Function Families

Denote by $\mathcal{N}$ the set of all positive integers, n the security parameter, Σ the alphabet $\{0, 1\}$ and $\#S$ the number of elements in a set S. By $x \in_R S$ we mean that x is chosen randomly and uniformly from the set S. The composition of two functions f and g is defined as $f \circ g(x) = f(g(x))$. Throughout the paper l and m

Manuscript received May 1, 1992.
Manuscript revised August 10, 1992.

† The authors are with the Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Wollongong, NSW 2522, Australia.
†† The author is with ATR Communication Systems Research Laboratories, Kyoto-fu, 619-02 Japan.

will be used to denote polynomials from $\mathcal{N}$ to $\mathcal{N}$. First we give our formal definition of one-way functions.

Definition 1: Let $f: D \to R$ be a polynomial time computable function, where $D=\bigcup_n \Sigma^{l(n)}$ and $R=\bigcup_n \Sigma^{m(n)}$. f is a one-way function if for each probabilistic polynomial time algorithm M, for each polynomial Q and for all sufficiently large n, $\Pr\{f_n(x)=f_n(M(f_n(x)))\}<1/Q(n)$, where $x \in_R \Sigma^{l(n)}$ and f_n denotes the restriction of f on $\Sigma^{l(n)}$.

Let $F=\{F_n | n \in \mathcal{N}\}$ be an infinite family of functions, where $F_n=\{f | f: \Sigma^{l(n)} \to \Sigma^{m(n)}\}$. Call F a function family mapping $l(n)$-bit input to $m(n)$-bit output strings. F is *polynomial time computable* if there is a polynomial time algorithm (in n) computing all $f \in F$, and *samplable* if there is a probabilistic polynomial time algorithm that on input $n \in \mathcal{N}$ outputs uniformly at random a description of $f \in F_n$. (Note: following the tradition in the field, when the security parameter n is an input to an algorithm, it will actually be represented by the all-1 string $1^n \in \Sigma^n$.) In addition, we call F a *one-way* family of functions if the function f defined by $f_n \in_R F_n$ is a one-way function.

Now we introduce the definition of pseudo-random function families[5] which will be applied in Sect. 4.2. Intuitively, $F=\{F_n | n \in \mathcal{N}\}$ is a pseudo-random function family if to a probabilistic polynomial time Turing machine (algorithm), the output of a function f chosen randomly and uniformly from F_n, whose description is unknown to the Turing machine, appears to be totally uncorrelated to the input of f, even if the algorithm can choose input for f. The formal definition is described in terms of *(uniform) statistical tests for functions*. A (uniform) statistical test for functions is a probabilistic polynomial time Turing machine T that, given n as input and access to an oracle O_f for a function $f: \Sigma^{l(n)} \to \Sigma^{m(n)}$, outputs a bit 0 or 1. T can query the oracle only by writing on a special tape some $x \in \Sigma^{l(n)}$ and will read the oracle answer $f(x)$ on a separate answer-tape. The oracle prints its answer in one step.

Definition 2: Let $F=\{F_n | n \in \mathcal{N}\}$ be an infinite family of functions, where $F_n=\{f | f: \Sigma^{l(n)} \to \Sigma^{m(n)}\}$. Assume that F is both polynomial time computable and samplable. F is a pseudo-random function family iff for any statistical test T, for any polynomial Q, and for all sufficiently large n,

$$|p_n^f - p_n^r| < 1/Q(n),$$

where p_n^f denotes the probability that T outputs 1 on input n and access to an oracle O_f for $f \in_R F_n$ and p_n^r the probability that T outputs 1 on input n and access to an oracle O_r for a function r chosen randomly and uniformly from the set of all functions from $\Sigma^{l(n)}$ to $\Sigma^{m(n)}$. The probabilities are computed over all the possible choices of f, r and the internal coin tosses of T.

In Ref.(5), it has been shown that pseudo-random function families can be constructed from pseudo-random string generators. By the result of Ref.(10),(11), the existence of one-way functions is sufficient for the construction of pseudo-random function families.

2.2 Universal Hash Function Families

Universal hash function familes, first introduced in Ref.(3) and then developed in Ref.(18), play an essential role in many recent major results in cryptography and theoretical computer science. (See for example Refs.(10), (11), (16).) Let $U=\bigcup_n U_n$ be a family of functions mapping $l(n)$-bit input into $m(n)$-bit output strings. For two strings $x, y \in \Sigma^{l(n)}$ with $x \neq y$, we say that x and y collide with each other under $u \in U_n$ or x and y are siblings under $u \in U_n$, if $u(x)=u(y)$.

Definition 3: Let $U=\bigcup_n U_n$ be a family of functions that is polynomial time computable, samplable and maps $l(n)$-bit input into $m(n)$-bit output strings. Let $D_n=\{x \in \Sigma^{l(n)} | \exists u \in U_n \exists y \in \Sigma^{m(n)}$ such that $u(x)=y\}$ and $R_n=\{y \in \Sigma^{m(n)} | \exists u \in U_n, \exists x \in \Sigma^{l(n)}$ such that $y=u(x)\}$. Let $k \geqq 2$ be a positive integer. U is a (strongly) k-universal hash function family if for all n, for all k (distinct) strings $x_1, x_2, \cdots, x_k \in D_n$ and all k strings $y_1, y_2, \cdots, y_k \in R_n$, there are $\# U_n / (\# R_n)^k$ functions in U_n that map x_1 to y_1, x_2 to y_2, $\cdots$, and x_k to y_k.

An equivalent definition for the (strongly) k-universal hash function family is that for all k distinct strings $x_1, x_2, \cdots, x_k \in D_n$, when h is chosen uniformly at random from U_n, the concatenation of the k resultant strings $y_1=h(x_1)$, $y_2=h(x_2)$, $\cdots$, $y_k=h(x_k)$ is distributed randomly and uniformly over the k-fold Cartesian product R_n^k of R_n. The following *collision accessibility property* is a useful one.

Definition 4: Let $U=\bigcup_n U_n$ be a family of functions that is polynomial time computable, samplable and maps $l(n)$-bit input into $m(n)$-bit output strings. Let $k \geqq 1$ be a positive integer. U has the k-collision accessibility property, or simply the collision accessibility property, if for all n and for all $1 \leqq i \leqq k$, given a set $X=\{x_1, x_2, \cdots, x_i\}$ of i initial strings in $\Sigma^{l(n)}$, it is possible in probabilistic polynomial time to select randomly and uniformly functions from U_n^X, where $U_n^X \subset U_n$ is the set of all functions in U_n that map $x_1, x_2, \cdots$, and x_i to the same strings in $\Sigma^{m(n)}$.

k-universal hash function families with the collision accessibility property can be obtained from polynomials over finite fields.(3),(18) Denote by P_n the collection of all polynomials over $GF(2^{l(n)})$ with degrees less than k, i.e.,

$$P_n=\{a_0+a_1x+\cdots+a_{k-1}x^{k-1}|a_0, a_1, \cdots, a_{k-1} \in GF(2^{l(n)})\}.$$

For each $p\in P_n$, let u_p be the function obtained from p by chopping the first $l(n)-m(n)$-bits of the output of p whenever $l(n)\geqq m(n)$, or by appending a fixed $m(n)-l(n)$-bit string to the output of p whenever $l(n)<m(n)$. Let $U_n=\{u_p|p\in P_n\}$, and $U=\bigcup_n U_n$. Then U is a (strongly) k-universal hash function family, which maps $l(n)$-bit input into $m(n)$-bit output strings and has the collision accessibility property.

2. 3 Sibling Intractable Function Families

Let $k=k(n)$ be a polynomial with $k\geqq 1$. Let $H=\{H_n|n\in\mathcal{N}\}$, where $H_n=\{h|h: \Sigma^{l(n)}\to\Sigma^{m(n)}\}$, be an infinite family of functions that is one-way, polynomial time computable and samplable, and that has the collision accessibility property. Also let $X=\{x_1, x_2, \cdots, x_i\}$ be a set of i initial strings in $\Sigma^{l(n)}$, where $1\leqq i\leqq k$, and h be a function in H_n that maps $x_1, x_2, \cdots, x_i$ to the same string. Let F, called *a sibling finder*, be a probabilistic polynomial time algorithm that on input X and h, outputs either "?" ("I cannot find") or a string $x'\in\Sigma^{l(n)}$ such that $x'\notin X$ and $h(x')=h(x_1)=h(x_2)=\cdots=h(x_i)$. Informally, H is a k-sibling intractable *function family*, or k-SIFF for short, if for any $1\leqq i\leqq k$, for any sibling finder F, the probability that F outputs an x' is negligible. More precisely:

Definition 5 Let $k=k(n)$ be a polynomial with $k\geqq 1$. Let $H=\{H_n|n\in\mathcal{N}\}$, where $H_n=\{h|h: \Sigma^{l(n)}\to\Sigma^{m(n)}\}$, be a family of functions that is one-way, polynomial time computable and samplable, and that has the collision accessibility property. Also let $X=\{x_1, x_2,\cdots, x_i\}$ be any set of i initial strings, where $1\leqq i\leqq k$. H is a k-sibling intractable function family, or simply k-SIFF, if for each $1\leqq i\leqq k$, for each sibling finder F, for each polynomial Q, and for all sufficiently large n,

$$\Pr\{F(X, h)\neq ?\}<1/Q(n),$$

where h is chosen randomly and uniformly from $H_n^X\subset H_n$, the set of all functions in H_n that map $x_1, x_2, \cdots$, and x_i to the same strings in $\Sigma^{m(n)}$, and the probability $\Pr\{F(X, h)\neq ?\}$ is computed over H_n^X and the sample space of all finite strings of coin flips that F could have tossed.

Here are several remarks on SIFF which follow directly from the definition of SIFF:

1. If $H=\{H_n|n\in\mathcal{N}\}$ is a k-SIFF for some $k\geqq 1$, then the function f defined by $f_n\in_R H_n$ is a one-way function.
2. A one-way one-to-one function is a 1-SIFF.
3. A universal one-way hash function family introduced in Ref.(13) is a 1-SIFF.
4. If $H=\{H_n|n\in\mathcal{N}\}$ is a k-SIFF, then it is also an i-SIFF for any $1\leqq i<k$.

In the next section we give an explicit construction of SIFF from any one-way function.

3. Construction of SIFF

In Ref.(16), Rompel showed that universal one-way hash function families, that is, 1-SIFF, can be constructed from any one-way function. Rompel's result is the starting point of our construction of k-SIFF. The following theorem shows that 1-SIFF can be transformed into 2^s-SIFF for any $s=O(\log n)$. This result is general enough owing to the fact that a k-SIFF is also an i-SIFF for any $1\leqq i<k$.

Theorem 1: Let l, m' and m be polynomials with $m'(n)-m(n)=O(\log n)$. Let $k=2^{m'(n)-m(n)}$. Assume that $H'=\{H_n'|n\in\mathcal{N}\}$ is a 1-SIFF mapping $l(n)$-bit input to $m'(n)$-bit output strings, and $U=\{U_n|n\in\mathcal{N}\}$ a k-universal hash function family that has the collision accessibility property and maps $m'(n)$-bit input to $m(n)$-bit output strings. Let

$$H_n=\{u\circ h'|h'\in H_n', u\in U_n\}$$

and $H=\{H_n|n\in\mathcal{N}\}$. Then H is a k-SIFF mapping $l(n)$-bit input into $m(n)$-bit output strings.

Proof: First we observe that H has the collision accessibility property, simply because that H' is samplable and that U has the collision accessibility property.

Now assume for contradiction that there exists a sibling finder F that, for infinitely many n, on input some $X=\{x_1, x_2, \cdots, x_i\}$ and $h\in_R H_n^X$ where $1\leqq i\leqq k$, outputs with probability at least $1/Q(n)$ a string $x'\in\Sigma^{l(n)}$ such that $x'\notin X$ collides with all strings in X, where Q is a polynomial and H_n^X is the set of all functions in H_n that map $x_1, x_2, \cdots$, and x_i to the same strings in $\Sigma^{m(n)}$. We show a contradiction to the assumption that H' is a 1-SIFF. More specifically, we construct a probabilistic polynomial time algorithm M that uses F as an oracle and succeeds with probability $1/(2kQ(n))$ in either of the following two actions. The first action is to find a string colliding with x_j for some $1\leqq j\leqq i$, and the second action is, when $i<k$, to obtain the inverse of some string from $y_{i+1}, y_{i+2}, \cdots, y_k$ with respect to a function h' chosen uniformly at random from H_n', where each y_j, $i+1\leqq j\leqq k$, is generated by first picking randomly an element from $\Sigma^{l(n)}$ and then evaluating the function h' at the random point.

Given F, X, $\{y_{i+1}, y_{i+2}, \cdots, y_k\}$ and h', M runs according to the following steps:

1. Choose $z\in_R\Sigma^{m(n)}$.
2. Choose randomly $u\in U_n$ such that $u(y_1)=u(y_2)=\cdots=u(y_i)=u(y_{i+1})=\cdots=u(y_k)=z$, where $y_j=h'(x_j)$

for all $1 \leqq j \leqq i$.
3. Call F with X and $h = u \circ h'$ as input. Let the output of F be x'.

Note that for $h' \in_R H'_n$, the probability that $y_{j_1} = y_{j_2}$ for some $1 \leqq j_1 \neq j_2 \leqq k$ is negligible. Otherwise $H' = \{H'_n | n \in \mathcal{N}\}$ would not be a 1-SIFF. In the following discussion, we will assume that $y_1, y_2, \cdots, y_k$ are all distinct.

The function $h = u \circ h'$ is clearly a random element of H_n^X, as U is a k-universal hash function family with the collision accessibility property, and h', z and u are all chosen randomly. Denote by S_1 the set of all the siblings of $x_1, x_2, \cdots, x_i$ and by S_2 the set of the inverses of $y_{i+1}, y_{i+2}, \cdots, y_k$, both with respect to h'. Note that S_1 and S_2 are disjoint sets when $y_1, y_2, \cdots, y_k$ are all distinct, and that $x' \neq ?$ iff $x' \in S_1$ or $x' \in S_2$. Therefore, we have

$$\Pr\{x' \neq ?\} = \Pr\{x' \in S_1\} + \Pr\{x' \in S_2\}.$$

By assumption we have

$$\Pr\{x' \neq ?\} \geqq 1/Q(n).$$

This implies that either

$$\Pr\{x' \in S_1\} \geqq 1/2Q(n)$$

or (when $i < k$)

$$\Pr\{x' \in S_2\} \geqq 1/2Q(n).$$

$\Pr\{x' \in S_1\} \geqq 1/2Q(n)$ implies that x' collides, with probability at least $1/(2iQ(n)) \geqq 1/(2kQ(n))$, with x_j for some $1 \leqq j \leqq i$ under the randomly chosen function h'. This contradicts our assumption that H' is a 1-SIFF. On the other hand, when $i < k$, $\Pr\{x' \in S_2\} \geqq 1/2Q(n)$ implies that with probability at least $1/(2(k-i)Q(n)) \geqq 1/(2kQ(n))$, x' is the inverse of y_j for some $i+1 \leqq j \leqq k$ with respect to $h' \in_R H'_n$, which contradicts the fact that if H' is a 1-SIFF then the function defined by choosing $h' \in_R H'_n$ is a one-way function. In summary, $\Pr\{x' \neq ?\} \geqq 1/Q(n)$ is a contradiction to the assumption that H' is a 1-SIFF. This completes the proof. □

Combining Theorem 1 with Rompel's result that universal one-way hash function families, i.e., 1-SIFF, can be obtained from any one-way function, and with the fact that a 2^s-SIFF is also an i-SIFF for all $1 \leqq i < 2^s$, we have:

Theorem 2: k-SIFF can be constructed from any one-way function.

In the following section we will apply the results of this section to the hierarchical access control problem and provide a solution based on SIFF.

4. The Hierarchical Access Control Problem

In today's modern society, hierarchical structures exist in various forms, from business corporations to government departments, each resembling a directed graph (such as a tree) in its figurative shape, with a particular node of the graph the highest point of command. In mathematical terms, such a hierarchy usually take the form of a *partially ordered set* with the highest point of command being the *maximal node*. The various positions throughout the hierarchical structure are then represented as internal nodes, each being the point of command over its underlying sub-graph, consisting also of nodes.

In a hierarchical organization which deals with some amount of sensitive information, the security of certain pieces of information must often be maintained at a certain level which corresponds to a particular depth in the hierarchical organization represented by the graph. A typical case would be that of a banking corporation where the manager deals with private and sensitive data. Such data should not be accessible to company members with positions and authority lower than the manager. However, the opposite condition is often required to be fulfilled. The manager should be able to access data belonging to employees working underneath him/her in the hierarchical organization.

In past years cryptography has often been used to ensure the security of sensitive data. Of more recent interest, however, is the problem of organizing cryptographic keys in a hierarchical manner to mirror the structure of the organization employing the cryptographic techniques for security. The problem of generating and updating keys for a hierarchical organization, called the *hierarchical access control problem*, was first posed by Akl and Taylor in 1982.(1) Since then many solutions or partial solutions to the problem have been proposed.(2),(4),(6),(8),(9),(12),(14) A common drawback with these schemes is that all of them are based on a single cryptographic assumption, that is the (supposed) difficulty of breaking the RSA cryptosystem,(15) and make heavy use of the underlying algebraic properties of the crypto-function.

In Ref.(17), Sandhu gave a solution to the *special* case when an organization has a tree structure, using a set of one-way functions. However, the problem of solving the general case of partially ordered sets under the weakest assumption of the existence of one-way functions, remains an interesting open problem. In this section we give a simple solution to the open problem. Incorporated into our solution are the idea of Sandhu for the tree structure and an elegant use of SIFF. A remarkable feature of our solution is that each node in the hierarchical structure needs to keep only one secret key.

The problem of access control in hierarchical organizations is described more formally in Sect. 4.1. A definition for security of key generation schemes is introduced in the same section. This is followed by a detailed description of the key generation scheme and a proof of its security in Sect. 4.2 and Sect. 4.3 respectively. An improvement of the scheme is presented in

Sect. 4.4 and several issues on updating keys are briefly discussed in Sect. 4.5.

4. 1 Preliminaries

Usually, an organization G consists of a set of $P(n)$ members together with a hierarchical relation among the members, here P is a polynomial and the computational power of all members in the organization is bounded by probabilistic polynomial time. Such a hierarchical organization can be well modeled by an algebraic system called a *partially ordered set*. Let $S=\{N_1, N_2, \cdots, N_{P\cdot(n)}\}$ be a set of $P(n)$ nodes, each of which represents a member of the organization. Denote by $\geqq$ the hierarchical relation within the organization. Then G is determined by the pair of S and $\geqq$. In mathematical terms, the organization G is called a partially ordered set or *poset* for short. For convenience, in the following discussions we will sometimes interchange the terms (*hierarchical*) *organization* and *poset*, and the terms *member* and *node*.

Every poset has some nodes called *maximal nodes*. Each maximal node N_i has the property that there is no node $N_j \in S$ such that $N_j \geqq N_i$ and $N_j \neq N_i$. In this paper, we will only be concerned with such a hierarchical organization that has only one *maximal node* N_0. Results in this paper can be readily generalized to the case where a hierarchical organization has multiple maximal nodes. Assume that N_i and N_j are two different nodes in S. N_i is called an *ancestor* of N_j (or equivalently, N_j is a *descendant* of N_i) if $N_i \geqq N_j$. N_i is called a *parent* of N_j (or equivalently, N_j is a *child* of N_i) if N_i is an ancestor of N_j and there is no other node $N_k \in S$ with $N_i \geqq N_k \geqq N_j$. Now assume that $S' \subset S$ is a subset of S. S' induces a sub-poset that consists of the set $\Theta(S')$ and the partial order relation $\geqq$, where $\Theta(S')$ consists of both the nodes in S' and the nodes which are descendants of nodes in S'.

A *Hasse diagram* of an organization is a figure consisting of nodes, with an arrow directed downwards from N_i to N_j whenever N_i is a parent of N_j. As the correspondence between an organization and its Hasse diagram is obvious, in the following discussions we will not distinguish between an organization and its Hasse diagram. In particular, we will not distinguish between a node in S and the member of the organization represented by the node.

The *hierarchical access control problem* for an organization G essentially reduces to the problem of generating a key K_i for each node N_i in such a way that for any nodes N_i and N_j, the node N_i is able to derive from K_i the key K_j of N_j iff $N_i \geqq N_j$. Related to this is the problem of key updates of the nodes. Key updating is required when the structure of the organization is modified. Typical changes to the structure includes the deletion and addition of nodes. Key updating is also required when some keys are lost or when the duration of validity of the keys has expired.

Next we discuss the definition of security of a key generation scheme for a hierarchical organization. Any key generation scheme should at least fulfill the requirement that it is computationally difficult for members of the organization, represented by a subset S' of S, to *find* by collaboration the key K_i of a node N_i not in $\Theta(S')$, where $\Theta(S')$ consists of both the nodes in S' and the nodes which are descendants of nodes in S'. When N_i is an internal node or the maximal node N_0, which implies that N_i has at least one child, the following more general requirement should be fulfilled. That is, it is computationally difficult for S' to *simulate* N_i's procedure for generating the key of a child of N_i. Note that S' may or may not be able to find the key of N_i and that the child of N_i may or may not be in $\Theta(S')$. The reason for considering the general requirement is that N_i's procedure for generating the key of the child, even if the child is in $\Theta(S')$, is a privilege of N_i, and the privilege should not be shared by any other node that is not an ancestor of N_i. The following is a formal definition of security of a key generation scheme.

Definition 6: Let G be a hierarchical organization with $P(n)$ nodes (members). Denote by S the set of the $P(n)$ nodes. A key generation scheme for a hierarchical organization is secure if for any $S' \subset S$, for any node $N_i \notin \Theta(S')$, for any polynomial Q and for all sufficiently large n, the probability that the nodes in S' are able to find by collaboration the key K_i of the node N_i whenever N_i has no child, or to simulate N_i's procedure for generating the key of a child of N_i whenever N_i is an internal node or the maximal node N_0, is less than $1/Q(n)$.

4. 2 Key Generation

Denote by ID_i the identity of the node N_i. Assume that every ID_i can be described by an $l(n)$-bit string, where l is a polynomial. Let $F=\{F_n | n \in \mathcal{N}\}$ be a pseudo-random function family, where $F_n=\{f_K | f_K: \Sigma^{l(n)} \rightarrow \Sigma^n, K \in \Sigma^n\}$ and each function $f_K \in F_n$ is specified by an n-bit string K. Let $H=\{H_n | n \in \mathcal{N}\}$ be a k-SIFF mapping n-bit input to n-bit output strings. Also assume that k is sufficiently large so that no nodes could have more than k parents. The following key generation procedure can be done either by a trusted third party or by the maximal node itself.

First a random string $K_0 \in_R \Sigma^n$ is chosen for the maximal node N_0. For the nodes without a key, either with one or more parents, the following two steps should be completed until all the nodes in S have been assigled keys.

1. Nodes with a single parent

Given a node N_i with its parent N_j which has already been assigned a key K_j, the key to be assigned to N_i is the n-bit string

$$K_i = f_{K_j}(ID_i) \tag{1}$$

2. Nodes with two or more parents

Given a node N_i with all its p parents $N_{j1}, N_{j2}, \cdots, N_{jp}$ having been assigned keys $K_{j1}, K_{j2}, \cdots, K_{jp}$, the key K_i for N_i is chosen as a random string $K_i \in_R \Sigma^n$. From H_n a function h_i is chosen randomly and uniformly such that $f_{K_{j1}}(ID_i), f_{K_{j2}}(ID_i), \cdots, f_{K_{jp}}(ID_i)$ are mapped to K_i. That is,

$$h_i(f_{K_{j1}}(ID_i)) = h_i(f_{K_{j2}}(ID_i)) = \cdots$$
$$= h_i(f_{K_{jp}}(ID_i)) = K_i \tag{2}$$

The function h_i is then made public, making all the ancestors of N_i aware of h_i.

It is clear that if $N_j \geq N_i$, then K_i can be derived from K_j. When N_j is a parent of N_i, K_i can be computed via either $K_i = f_{K_j}(ID_i)$ if N_j is the single parent of N_i, or $K_i = h_i(f_{K_j}(ID_i))$ if N_i has other parents. When N_j is not a parent of N_i, all keys in the path from N_j to N_i are computed downwards and K_i is obtained in the final stage.

4.3 Security of the Key Generation Scheme

This section proves that the key generation scheme is secure. A corollary of the result is that secure key generation schemes for hierarchical organizations can be obtained from any one-way function. We will only provide the sketch of the proof for the security, as it is a relatively straightforward procedure to translate the less formal proof into a formal one.

Theorem 3: The key generation scheme for a hierarchical organization is secure.

Proof (Sketch): Let S' be a subset of S, i.e. $S' \subset S$, and let $\Theta(S')$ be the set of the descendants of nodes in S' plus the nodes in S'. Also let N_i be a node not in $\Theta(S')$, i.e. $N_i \notin \Theta(S')$. According to the definition for security (Definition 6), we need to consider the following two cases:

Case 1: N_i has no child and S' can directly find the key K_i of N_i.

Case 2: N_i has one or more children and S' can simulate N_i's procedure for generating the key K_j of some child N_j of N_i.

First we discuss Case 1 where N_i has no child. Note that the key K_i of N_i is constrained either by the Eq. (1) when N_i has only a single parent, or by the Eq. (2) when N_i has two or more parents. In other words, K_i is derived from the key(s) of the parent(s) of N_i by the use of the pseudo-random function family. Therefore, obtaining K_i by S' implies that S' is able to predict the output of the pseudo-random function family, which is a contradiction.

Now we consider Case 2 where N_i is an internal node or the maximal node N_0, and S' can simulate N_i's procedure for generating the key K_j of some child N_j of N_i. Note that N_j may or may not be a member of $\Theta(S')$. For our key generation scheme, being able to simulate N_i's procedure for generating the key K_j of the child N_j of N_i implies being able to get either K_i when N_i is the single parent of N_j, or $f_{K_i}(ID_j)$ when N_j has other parents than N_i. Also note that getting K_i or $f_{K_i}(ID_j)$ means getting the keys of all the descendants of N_j besides the key K_j of N_j. Thus there are only two situations to be considered when S' is able to get K_i or $f_{K_i}(ID_j)$ but fails to mimic any of the parents of N_i. These two situations are:

Situation 1: N_i is an ancestor of some node(s) in $\Theta(S')$.

Situation 2: N_i is not the ancestor of any node in $\Theta(S')$.

Consider Situation 1 first. Since N_i is an ancestor of a node in $\Theta(S')$, there is a path from N_i to the node in $\Theta(S')$. N_i can derive the key of the node in $\Theta(S')$ by evaluating the pseudo-random function family and (instances of) the sibling intractable function family which appear in the path. Therefore, getting the key K_i of N_i or $f_{K_i}(ID_j)$ by S' implies that S' can do at least one of the following three actions: invert the pseudo-random function family, find a collision string for (instances of) the sibling intractable function family (appearing in the path from N_i to the node in $\Theta(S')$), or invert (instances of) the sibling intractable function family. The success of any of these actions with a high probability is a contradiction. Compared to Situation 1, Situation 2 is easier to analyze. Since N_i is not the ancestor of any node in $\Theta(S')$, there is no path from a node in $\Theta(S')$ to N_i. Thus getting K_i or $f_{K_i}(ID_j)$ by $\Theta(S')$ implies that $\Theta(S')$ can predict the output of the pseudo-random function family. This is also a contradiction.

4.4 Improvement of the Key Generation Scheme

A problem with the above key generation scheme is that a node must pass through a number of intermediate descendants in order to arrive at a given distant (non-child) descendant node. This may be an inconvenience to the members of the hierarchical organization. This traversal down the structure requires the use of the sibling intractable function family and the pseudo-random function associated with each node along the traversed path in order to find the keys of the intermediate nodes.

A solution to the problem consists of a modification to the key generation phase. For a given node N_i with q ancestors (*including the parents*) N_{i1}, N_{j2}, $\cdots$, and N_{jq}, the generation of the key of N_i involves the selection of a random string $K_i \in_R \Sigma^n$ and the selection of $h_i \in H_n$ randomly and uniformly such that $f_{K_{j1}}(ID_i)$, $f_{K_{j2}}(ID_i)$, $\cdots$, and $f_{K_{jq}}(ID_i)$ are all mapped to K_i:

$$h_i(f_{K_{j1}}(ID_i)) = h_i(f_{K_{j2}}(ID_i)) = \cdots$$

$$=h_i(f_{K_{jq}}(ID_i))=K_i \qquad (3)$$

In this way any ancestor of N_i which was involved in the generation of K_i can access N_i directly without the need to pass any intermediate nodes. An extensive treatment of this issue together with other solutions to the hierarchical access control problem is presented in Ref. (19).

4. 5 Key Updating

It is natural to expect that the structure of an organization (i.e., the shape of the corresponding Hasse diagram) will change throughout time, and thus the keys of the nodes will also need to be updated. Some typical changes include the addition and deletion of nodes, and the establishment and removal of links between nodes. Another reason for the renewal of the keys is the replacement of one member of the organization with another, without involving any change to the structure (Hasse diagram) itself. The arrival of a new member to the organization implies the creation of a new identity information for that member. Key updating is also required when some keys are lost or when the duration of validity of some keys has expired. In this section we will consider briefly three of the most typical problems related to the maintenance of the internal nodes of the hierarchical structure. These are the addition and deletion of nodes, and the replacement of the identification information of nodes. The case of leaf nodes is trivial and will not be discussed.

4. 5. 1 Addition and Deletion of Nodes

When nodes are added or deleted there are a number of possibilities as to how the subposets affected by the change should be maintained. When a new node N_k is added between node N_j and its parent node N_i, N_k becomes the new parent of N_j, and N_i the parent of N_k. The descendants of N_k which includes N_j are effectively shifted one level down in the organization (Hasse diagram). It is clear that the addition of a new node followed by a shift down of all its descendants requires the generation of new keys for that node and its descendants. Only the new node requires a new identity information.

In the case of the deletion of a node, its descendants becomes the descendants of its parent(s) and new keys must be generated for the descendants. This corresponds to an upward shift by one level of these descendants in the organization.

4. 5. 2 Replacement of an Identity

The replacement of the identity information of a node can be due to a number of changes in the organization. A member at a node can be replaced by another current member or by a new member from outside the organization. Often, the identity of a node simply needs to be changed following some organizational decision. In all these cases the keys for all the descendants of that node need to be generated again. The node itself is assigned a new key which is used to generate and assign the keys of its children nodes.

5. Other Applications of SIFF

There are numerous ways that SIFF can be applied. In the following, three other applications of SIFF are briefly discussed. The first application is to the shared mail box problem, the second to the access control in distributed systems, and the third to the multiple message authentication problem. A further application of SIFF to database authentication is presented in Ref. (7).

5. 1 The Shared Mail Box Problem

Suppose that there is a group consisting of k users $U_1, U_2, \cdots, U_k$. Each user U_i has a private mail box B_i. Assume also that there is a shared mail box B_S. The shared mail box problem consists of the design of a cryptographic system for the group that has the following features:

1. Each user U_i holds just one secret key K_i of n bits, for some $n \in \mathcal{N}$.
2. For each $1 \leqq i \leqq k$, the mail box B_i can only be opened by the user U_i who possesses the secret key K_i.
3. The shared mail box B_S can be opened by every user in the group, but not by any outsider.
4. Even when $k-1$ users conspire together, it is computationally difficult for the $k-1$ users to open the other user's private mail box.

This problem represents an abstraction of many practical applications where the determination of access rights to various resources is required. The traditional way for solving the access problem with the private and shared mail boxes is to let each user hold two keys, one for his or her private mail box and the other for the shared mail box. The traditional solution becomes very impractical when the user is a member of a number of different groups, and hence has to hold as many keys for shared mail boxes as the total number of groups he or she belongs to. We present a simple solution to the problem by the use of a SIFF and a secure secret-key block cipher, both of which can be constructed from any one-way function. First of all, we, a trusted third party, choose a secure secret-key block cipher for the purpose of locking the private and shared mail boxes. Then we choose a $(k-1)$-SIFF $H=\{H_n | n \in \mathcal{N}\}$ that maps n-bit input to n-bit output strings. The following is the key generation procedure for the group.

• Choose for each user U_i a random n-bit string K_i as his or her secret key.
• Choose a random n-bit string K_s for the shared mail box B_s.
• Select from H_n a random function h such that all private keys $K_1, K_2, \cdots, K_k$ are mapped to K_s, i.e., $h(K_1)=h(K_2)=\cdots=h(K_k)=K_s$. Then make h public.

It is easy to see that the scheme fulfills all the four requirements. In particular, each user U_i can apply the secure block cipher with K_i as a key to open or lock his or her private mail box B_i, and with $h(K_i)$, which is mapped to K_s, as a key to open or lock the shared mail box B_s.

5.2 Access Control in Distributed Systems

SIFF can also be used to control access of data in a distributed system which are geographically dispersed. Each site in the system would have two levels of access, the first being applied to the site as a whole, while the second to control access to resources and data stored at that site.

In the first access level, the access by one site to another is determined using SIFF. Hence, a given site can determine which other sites that may have access to it. This, in effect, classifies sites according to their sensitivity, and may be governed by various performance and practical necessities. In the second access level, which assumes the granting of access in the first level, transactions that access multilevel resources and data can be controlled also using SIFF.

5.3 The Multiple Message Authentication Problem

Message authentication is an important part of information security. A common method is to append to a message to be authenticated a short tag such as a checksum, by using a modification detection code. In some cases, we have many (independent) messages to be authenticated. Two usual methods are for each message to be given a tag independent of one another, and for the concatenation of all the messages to be given a single common tag. In the first method the resulting number of tags may prove too impractical to be maintained, while in the second method the validation of one message requires the use of all other (unrelated) messages in the re-calculation of the tag.

A preferred method would be one that employs a single common tag for all the messages in such a way that a message can be verified individually without involving other messages. This can be achieved by using SIFF in which all messages are represented as strings of $l(n)$-bit long (with padding if required). The common tag is $m(n)$-bits long and the integer k must be larger than the number of all messages for which the common tag is to be created. A k-SIFF $H=\{H_n | n\in\mathcal{N}\}$ is then chosen that maps $l(n)$-bit input to $m(n)$-bit output strings. In general the application of the k-SIFF is restricted to those situations that require a common tag for all input messages. Two conceivable simple situations that may arise and that may gain advantage from using a common tag instead of individual tags are the following.

The first situation is the case where a software company produces a large number of software products which can function together as an integrated system or which can function independently as a stand-alone program. A customer may purchase any single component, and later gradually purchase the entire collection of components to create the integrated system. The customer must also be satisfied that the products being bought are authentic from the software company and have not been tampered with illegally or infected by computer viruses. In such a situation it may be preferable if the company issues a common tag t and function h for all its products to a customer when that customer makes a first-time purchase. For subsequent purchases the company need only send the newly-purchased component without the need to generate or send an accompanying individual tag for that component. The customer can verify the authenticity of each component using t and h.

In order to achieve this effect the company must first choose a random $m(n)$-bit tag t. All softwares to be purchased are assumed to be encrypted for their security during transportation. The company then chooses randomly and uniformly from H_n a function h that maps all the encrypted softwares of the company and the cryptographic keys to the same tag t. For a first-time purchase, the company issues the tag t and the function h in a secure manner to the customer. For subsequent purchases the company can send the product to the customer over insecure communications line since any corruption to the encrypted product during its transit will be detected by the customer. Note that the tag t and the function h need not be maintained secret. They can in fact be published publicly, which then allows any party to authenticate any (encrypted) software from the company. This basic form of authenticating a product only satisfies the requirement that the customer receives the product in its original form. The approach can be somewhat augmented by providing each customer with a different tag t and function h. The approach does not address the issue of software piracy by customers.

The second situation is related to the first, and it concerns the authentication of a set of software components by a Trusted Verification and Certification Authority (TVCA). In this situation a group of competing companies agree to produce a given integrated system where each company produces one or more of the components of the system. After the companies complete their products, the TVCA evaluates the trustworthiness of each component and the trustworthiness of the integrated system as a whole. When the TVCA is satisfied it provides a ratings for each component and for the total integrated system. The TVCA then chooses randomly a tag t and chooses randomly and uniformly from H_n a function h that maps all the certified components to the tag. Customers that seek a trustworthy system can purchase the versions of the components prescribed by the TVCA and check the authenticity of each component by using t and h. This approach gives some guarantee to the customer that the components being purchased are indeed those verified and rated by the TVCA. This approach also prevents one company from undermining the integrity of second company by producing different versions of a component that reduces the security level achieved by the components of the second company. Each subsequent versions of components must be re-evaluated by the TVCA who repeats the above process of authenticating the products.

In general only certain situations warrant the use of a k-SIFF, and such cases must be evaluated from the points of view of security, the required computation and the convenience to the users. The above examples show that SIFF has potential for many areas of application.

6. Conclusion and Further Work

We have introduced the notion of SIFF which includes as a special case the notion of the universal one-way hash function family defined Naor and Yung in Ref.(13). We have also shown a simple method for transforming any universal one-way hash function family into a SIFF. Putting together this and Rompel's results, we have obtained the theoretically optimal result that SIFF can be constructed from any one-way function. To illustrate the potentials of SIFF we have presented a key generation scheme for hierarchical organizations, and suggested solutions to the shared mail box problem, access control in distributed systems and the multiple message authentication problem.

The applicability of a solution based on SIFF to a practical problem is largely determined by the compactness of SIFF. An improvement in the compactness of SIFF results directly in the improvement in the efficiency of a solution based on SIFF. For the construction of k-SIFF presented in Theorem 1, the length of a description of a function in H_n is of order $O(L_1(n)+L_2(n))$, where $L_1(n)$ and $L_2(n)$ are the lengths of a description of a function in U_n and in H'_n respectively. Searching for more compact constructions of SIFF from one-way functions, together with other applications of SIFF, is an interesting subject for further research.

Acknowledgments

The authors are grateful to Prof. Jennifer Seberry for continuous interest and support, and to Babak Sadeghiyan and other members of the Centre for Communications and Computer Security Research (Wollongong University, Australia) for helpful discussions and comments. Special thanks go to Toshiya Itoh for pointing out crucial errors in early versions of this paper. This work was supported in part by Telecom Australia under the contract number 7027 and by the Australian Research Council under the reference number A48830241.

References

(1) Akl, S. G. and Talor, P. D., "Cryptographic solution to a multilevel security problem," In *Advances in Cryptology-Proceedings of Crypto'82* (Santa Barbara, Aug. 1982), D. Chaum, R. L. Rivest, and A. T. Sherman, Eds., Plenum Press, NY, pp. 237-250.

(2) Akl, S. G. and Taylor, P. D., "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transactions on Computer Systems*, vol. 1, no. 3, pp. 239-248, 1983.

(3) Carter, J. and Wegman, M., "Universal classes of hash functions," J. Comput. & Syst. Sci., vol. 18, pp. 143-154, 1979.

(4) Chick, G. C. and Tavares, S. E., "Flexible access control with master keys," in *Advances in Cryptolcgy-Proceedings of Crypto'89*, Lecture Notes in Computer Scince, vol. 435 (1990), G. Brassard, Ed., Springer-Verlag, pp. 316-322.

(5) Goldreich, O., Goldwasser, S. and Micali, S., "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792-807, 1986.

(6) Hardjono, T. and Seberry, J., "A multilevel encryption scheme for database security," *in Proceedings of the 12th Australian Computer Science Conference,* pp. 209-218, 1989.

(7) Hardjono, T. Zheng, Y. and Seberry, J., "A new approach to database authentication," in *Proceedings of the Third Australian Database Conference* (Database '92) 1992.

(8) Harn, L. and Kiesler, T., "Authentication group key distribution scheme for a large distributed network," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 300-309, 1989.

(9) Harn, L. and Lin, H. -Y., "A cryptographic key generation scheme for multilevel data security," *Computer & Security*, vol. 9, no. 6, pp. 539-546, 1990.

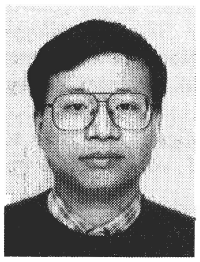
(10) Håstad, J., "Pseudo-random generation under uniform assumptions," in *Proceedings of the 22-nd ACM Symposium on Theory of Computing*, pp. 395-404, 1990.

(11) Impagliazzo, R., Levin, L. and Luby, M., "Pseudo-random generation from one-way functions," in *Proceedings of the 21-st ACM Symposium on Theory of Computing*, pp. 12-24, 1989.

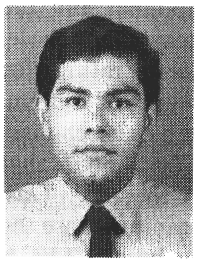
(12) MacKinnon, S. J., Taylor, P. D., Meijer, H. and Akl, S. G., "An optimal algorithm for assigning cryptographic keys to access control in a hierarchy," *IEEE Trans. Comput.*, vol. C-34, no. 9 pp. 797-802, 1985.

(13) Naor, M. and Yung, M., "Universal one-way hash functions and their cryptographic applications," in *Proceed-*

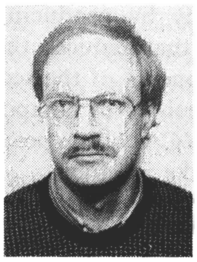
ings of the 21-st ACM Symposium on Theory of Computing, pp. 33-43, 1989.
(14) Ohta, K., Okamoto, T. and Koyama, K., "Membership authentication for hierarchical multigroup using the extended Fiat-Shamir scheme," in *Advances in Cryptology-Proceedings of Euro Crypt'90*, Lecture Notes in Computer Science, vol. 473 (1991), I. B. Damgård, Ed., Springer-Verlag.
(15) Rivest, R. L., Shamir, A. and Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2 pp. 120-128, 1978.
(16) Rompel, J., "One-way functions are necessary and sufficient for secure signatures," in *Proceedings of the 22-nd ACM Symposium on Theory of Computing*, pp. 387-394, 1990.
(17) Sandhu, R. S., "Cryptographic implementation of a tree hierarchy for access control," *Inf. Process. Lett.*, vol. 27, no. 2, pp. 95-98, 1988.
(18) Wegman, M. and Carter, J., "New hash functions and their use in authentication and set equality," *J. Comput. & Syst. Sci.*, vol. 22, pp. 265-279, 1981.
(19) Zheng, Y. Hardjono, T., and Seberry, J., "New solutions to access control in a hierarchy," *Technical Report*, Dept. of computer science, University of Wollongong, 1992.



Yuliang Zheng received his B.S. degree in computer science from the Southeast University (formerly Nanjing Institute of Technology), Nanjing, China, in 1982, and the M.E. and Ph.D. degrees, both in electrical and computer engineering, from the Yokohama National University, Yokohama, Japan, in 1988 and 1991 respectively. From 1982 to 1984 he was with the Guangzhou Research Institute for Communications, Guangzhou (Canton), China, and from February 1991 to January 1992 he was a Post-Doctoral Fellow in the Department of Computer Science, The University of New South Wales (University College, Canberra), Australia. Since February 1992 he has been a Lecturer in the Department of Computer Science, The University of Wollongong, Australia. His current research interests includes computer network security, cryptography, computational complexity theory and information theory. Yuliang Zheng is a member of the IEEE and the IACR.



Thomas Hardjono obtained the B.Sc. (Hons) degree in Computer Science from The University of Sydney and the Ph.D. degree also in Computer Science from The University of New South Wales, Australia, in 1987 and 1991 respectively. He is currently with ATR Communication Systems Research Laboratories, Communication Software Division in Kyoto, Japan. His research interests include computer systems and database security, cryptography and distributed systems. Thomas Hardjono is a member of the ACM and the IEEE.



Josef Pieprzyk received the M.Sc. degree in Electrical Engineering from the Academy of Technology in Bydgoszcz, Poland and a second M.Sc. degree in Mathematics from University of Torun, Poland in 1972 and 1975, respectively. In 1980 he obtained the Ph.D. degree from the Polish Academy of Sciences in Warsaw. Josef Pieprzyk was a Tutor and later a Senior Tutor in the Department of Telecommunication, Academy of Technology (1972-1980), Poland. In 1980 he became Assistant Professor and until 1986 when he appointed Lecturer in the Basser Department of Computer Science, The University of Sydney, Australia. He was then appoined as a Senior Lecturer in 1987. At the start of 1988 he continued his position as Senior Lecturer in the Department of Computer Science, The University of New South Wales (University College, Canberra). Josef Pieprzyk became an Associate Professor in the Department of Computer Science, The University of Wollongong, Australia in 1992. Josef Pieprzyk's research interests includes Computer Security and the theory and applications of Cryptography. He has published more than one hundred papers in the area of cryptography. He is co-author of *Cryptography: An Introduction to Computer Security* with Jennifer Seberry, which is published by Prentice-Hall. Josef Piepruzyk is a member of the IACR.