AUTHENTICATION OF TRANSIT FLOWS AND K-SIBLINGS ONE-TIME SIGNATURE

Mohamed Al-Ibrahim
Center for Computer Security Research
University of Wollongong
Wollongong , NSW 2522, Australia
ibrahim@ieee.org

Josef Pieprzyk
Center for Advanced Computing - Algorithms and Cryptography
Computing Department
Macquaire University
Sydney, NSW 2109, Australia
josef@ics.mq.edu.au

Abstract We exploit the unique features of the k-sibling Intractable hashing method in presenting two authentication schemes. In the first scheme, we propose a method which enables intermediate nodes in IP communication networks to verify the authenticity of transit flows. While in the second scheme, we introduce a new one-time digital signatures scheme.

Keywords: Network security, k-sibling hashing, source authentication, one-time signatures

1. INTRODUCTION

There has been considerable interest in group-based applications over the last few years with the emergence of new sorts of communication modes such as multicast, concast and broadcast. There has also been a remarkable increase in real-time applications such as online video/audio streaming which have special quality of service requirements. As far as the security of these applications is concerned, new challenges in designing security protocols for these applications have arisen. Usually, these applications have special quality-of-service (QoS) requirements and the

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35612-9_23 security services should be performed within its limits. One of the important security services is source authentication. Typical authentication schemes such as digital signatures have both high computational and space overhead, and hence they do not fulfill the new requirements of these applications. On the other hand, Message Authentication Codes (MAC) are more efficient, but does not provide non-repudiation service. Therefore, new techniques are required which not only can guarantee secure communication, but also maintains the efficiency of the application. This problem has been well defined and explored in the literature and several techniques have been proposed [2, 1, 3, 5, 14].

In this paper we continue the work in the direction of improving and developing efficient cryptography solutions for the problem of authentication in network communication. We introduce new authentication schemes that are based on the idea of the k-sibling intractable function family SIFF [8]. SIFF is a generalization of the universal one-way function family theorem (see also [15]). It has the property that given a set of initial strings colliding with one another, it is infeasible to find another string that would collide with the initial strings. This cryptographic concept has many useful applications in the security and we have exploit it to develop new authentication scheme. In this work, we start by expanding the idea of SIFF to hierarchical SIFF. Then, we proposed a scheme for authenticating 'transit' flows in IP communication. To our best knowledge, this topic has not been discussed elsewhere. Further, we propose a new one-time signature scheme that is efficient in generation and verification of signatures and with minimum space overhead, which is suitable for end-to-end real-time applications.

The paper is structured as follows. In the next section, we first illustrate the idea of k-SIFF and then expand it into a Hierarchical k-SIFF. In section 3, a scheme for authenticating transit flow in communication networks is illustrated. In section 4, the k-sibling one-time signature is presented.

2. K-SIBLING INTRACTABLE HASHING

The construction and security properties of k-sibling intractable hash functions are discussed in [8]. Briefly, let U= $\cup_n U_n$ be a family of functions mapping $l(n)$ bit into $m(n)$ bit output strings. For two strings $x, y \in \sum^{l(n)}$ were $x \neq y$, we say that x and y collide with each other under $u \in U_n$, or x and y are siblings under $u \in \mathcal{U}_n$, if $u(x) = u(y)$.

in other words, sibling intractable hashing provides a hashing that collides for k messages selected by the designer. It can be seen as the concatenation of two functions: universal hash function and collision resistant hash function. More formally, we say that a family of universal hash functions

$$\mathcal{U} = \{U_n : n = \mathcal{N}\}$$

holds k-collision accessibility property if for a collection $\mathcal{X} = \{x_1, \ldots, x_k\}$ of k random input values $U_n(x_1) = \ldots = U_n(x_k)$ where $U_n : \{0,1\}^{\ell(n)} \to \{0,1\}^{L(n)}$ and $\ell(n), L(n)$ are two polynomials in n (n is the security parameter and $\mathcal{N}$ is the set of all natural numbers). A family of collision resistant hash functions

$$\mathcal{H} = \{H_n : n = \mathcal{N}\}$$

consists of functions that are one-way, and finding any pair of colliding messages is computationally intractable. k-sibling intractable hash functions can be constructed as

$$kH_n = \{h \circ u : h \in H_n, u \in U_n\}$$

where $U_n : \{0,1\}^{\ell(n)} \to \{0,1\}^{L(n)}$ and $H_n : \{0,1\}^* \to \{0,1\}^{L(n)}$ (the notation $\{0,1\}^*$ stands for strings of arbitrary length). U_n is a collection of polynomials over $GF(2^{\ell(n)})$ of degree k.

2.1 Hashing with a Single Polynomial

The designer of a k-sibling intractable hash function first takes an instance of a collision intractable hash $H : \{0,1\}^* \to \{0,1\}^{\ell}$ (such as SHA1) and a collection of k messages $\{m_1, \ldots, m_k\}$ that are to collide. Next she computes

$$x_i = H(m_i)$$

randomly chooses $\alpha \in GF(2^{\ell})$ and determines a polynomial $U : \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ such that

$$U(x_i) = \alpha \text{ for } i = 1, 2, \ldots, k.$$

This can be done using the Lagrange interpolation. Having k points (x_i, α); $i = 1, \ldots, k$, it is easy to determine such a polynomial $U(x)$ and different from a straight line (see Appendix A). Note that, $k + 1$ points are needed to determine a polynomial of degree k.

Denote $\mathbf{H} = U \circ H$. By construction $\mathbf{H}(m_i) = \alpha$ for all $i = 1, 2, \ldots, k$. The hash function $\mathbf{H}$ can be characterized by the following properties:

- finding collisions (those incorporated by the designer in U as well as those existing in H) is computationally intractable assuming the attacker knows descriptions of two functions H and U. The descriptions must be available in a public, read-only registry. Note that the description of U takes $k + 1$ values from $GF(2^{\ell})$.

- the hash function treats messages as an unordered collection of elements. To introduce an order, the designer needs to calculate $\mathbf{H}$ for an ordered sequence of messages so any message $m_i = (i, m_i')$ where i indicates the position of the message in the sequence. In other words, $\mathbf{H}(i, m_i') = \alpha$ for all $i = 1, \ldots, k$,

Note that if the number of colliding messages is large (say $k > 1000$), then to compute hash values, one would need to fetch $k+1$ coefficients of polynomials $U(x)$. This introduces delays. Is there any other way to design k-sibling hashing?

2.2 Hierarchical Sibling Intractable Hashing

Given k-sibling intractable hash function $\mathbf{H}^{(k)}$ and a set $\mathcal{M} = (m_1, \ldots, m_{k^2})$ of messages. A k^2-sibling intractable hash function denoted as

$$\mathbf{H}^{(k^2)} = \mathbf{H}^{(k)} \circ \mathbf{H}^{(k)}$$

is a collection of $k+1$ k-sibling intractable hash functions where

$$\mathbf{H}_i = U_i \circ H \text{ with collisions in } \mathcal{M}_i = (m_{ik+1}, \ldots, m_{(i+1)k})$$

for $i = 0, \ldots, k-1$, and

$$\mathbf{H}_k = U_k \circ H \text{ with collisions in } \mathcal{X} = \{h_i = \mathbf{H}_i(\mathcal{M}_i); i = 1, \ldots, k\}.$$

To find hash value of a message, it is not necessary to know all polynomials U_i. For a message $m \in \mathcal{M}_i$, it is sufficient to know two polynomials only, namely, U_i and U_k.

In general, sibling intractable hashing with k^r colliding messages can be defined as

$$\mathbf{H}^{(k^r)} = \mathbf{H}^{(k)} \circ \mathbf{H}^{(k^{r-1})}$$

for $r > 2$. Similarly, to compute a hash value for a single message, it is necessary to learn r polynomials of degree k.

The polynomials $U_{i,j}(x)$ are in fact arranged in a tree structure. The leaves of the tree are $U_{1,j}$ for $j = 1, \ldots, k^{r-1}$. The next layer is created by polynomials $U_{2,j}$; $j = 1, \ldots, k^{r-2}$ and so on. The root is $U_{r,1}$.

Hierarchical sibling hashing holds the same security properties as the hashing with a single polynomial. The proof is relatively simple and follows the idea of Damgard's parallel hashing (see [7]).

3. AUTHENTICATION OF PACKETS

Message authentication is an important service in information security. Typical authentication schemes such as digital signatures uses public-keys, while Message Authentication Codes (MAC), uses private keys. Digital signatures are known for their high computation overhead, while MAC does not provide non-repudiation service. In cases, such as in IP communication, we may have a stream of independent messages to be authenticated. Neither typical digital signatures provides efficient solution, nor MAC provides enough security service. Therefore, new techniques are required to provide both security and efficiency.

The other motivation is the requirement by the intermediate nodes in IP network for a technique to authenticate the packets in their transitions from source destination. IPSec [12] is a security mechanism designed to provide security services for IP communication. It provides source authentication, confidentiality, as well as integrity. As far as source authentication is concerned, with the symmetric authentication option provided by IPsec, only hop-by-hop authentication can be achieved. This means that a node that receives a message only knows that it came from an authenticated node where they share common key in the domain. However, it would not be able for intermediate nodes along the path to check the authenticity of the messages. In doing this, it would be possible to discover harmful actions such as denial of service attack in their early stages, before it is propagated to the destination. We seek a mechanism that enables intermediate nodes to verify the source of the message.

Possible solutions that may used for this case are for each message to be given a tag independent of one another, or for the concatenation of all messages to be given a single common tag. In the first method, the resulting 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 recalculating the tag. A preferable method would be one that employs a single common tag for all the messages in a 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 a string of $l(n)$ bits long.

3.1 Authentication with Single Hashing

The security goal is to enable interested parties of the network (nodes) to authenticate messages (packets) in transit. A natural restriction imposed on authentication is that packets of the same message (generated by the same source) may travel through different routes. In effect, a node may see a small subset of all packets generated by the source. Those that are seen do not typically follow the initial order. Note that authentication of packets based on some sort of chaining is useless. Our solution is based on sibling intractable hashing.

Assumptions

- The source (sender) takes a message M and splits it into n packets (datagrams) In other words, $M = (m_1, \ldots, m_n)$ where m_i is the i-th datagram,
- There is a Public Key Infrastructure (PKI) that provides on demand authenticated public keys of all potential senders (normally in the form of certificates),
- The sender applies a secure signature scheme $SG = (S_{sk}, V_{pk}, G())$ for message authentication S_{sk} is the signing algorithm that for a given message m and a secret key sk produces a signature or $s = S_{sk}(m)$, V_{pk} is the verification algorithm that for a public key pk and a signature s returns 1 if the signature is valid and 0, otherwise. $G()$ generates a pair of keys: sk,pk. The meaning of "secure" will not be discussed here, and the interested reader can consult relevant papers (see [6]),
- The sender designs an instance of n-sibling intractable hash function $\mathbf{H}^{(n)}$ that is based on a collision intractable hash function H.

Sender

- Takes the sibling intractable hash $\mathbf{H}^{(n)}$ and computes the signature of the message M as

  $$s = S_{sk}(H(H^{(n)}(M)\|H(u_0, \ldots, u_n)), \text{timestamp})$$

  where $U(x) = u_0 + u_1x + \ldots + u_nx^n$ and $u_i \in GF(2^\ell)$,
- Puts the signature together with coefficients of $U(x)$ into a read-only registry R accessible to everybody.

Note that polynomial $U(x)$ must be used to produce the final hash value that is signed by the sender. This is done to prevent manipulation with the structure of the sibling intractable hash function $\mathbf{H}^{(n)}$.

Verifier

- Receives datagrams m_i where $i \in \{1, \ldots, n\}$,
- Contacts the registry R and fetches the signature and the coefficients u_i and recovers the polynomial $U(x)$,

- Obtains the authenticated public key pk of the sender from the PKI facility,
- Checks the validity of the signature using the algorithm $V_{pk}(s)$.

Note that the verification of the first datagram is the most expensive as it will take verification of signature (one exponentiation if signature is based on RSA or ElGamal) that comprises also the calculation of hash $h_i = H(m_i)$, computation of $U(h_i)$ and evaluation of $H(u_0, \ldots, u_n)$. Any new message can be verified using one extra evaluation of H and U. It computes the hash $h_i' = H(m_i)$ and computes $h' = U(h_i')$. If $h' = h$ then it accepts the message; otherwise, it rejects it. As far as communication is concerned, the verifier must fetch the signature and the polynomial $U(x)$. Note that the length of $U(x)$ is almost the same as the whole message M. This seems to be the weakest point of the construction.

3.2 Authentication with Hierarchical Hashing

In this case, the sibling intractable hashing is computed using a family of polynomials $U_{i,j}$ with $i = 1, \ldots, r$ and $j = 1, \ldots, k^{r-i}$. The message consists of $n = k^r$ datagrams. To compute $H^{(n)}(M)$ it is enough to fetch r polynomials of degree k (that is a sense, a path between a leaf and the root. If we choose k=2, then the verifier needs to fetch $3 \times \log_2 n$ coefficients. With pre-determined single points for the polynomials, the number can be reduced to $2 \times \log_2 n$ without security deterioration.

The tree of polynomials must also be subject to hashing (to make the verifier sure that she uses the correct instance of the sibling intractable hash). One good feature is that the verifier would like to use explicitly all polynomials she has imported from R. The signer may help the verifier by first using parallel hashing for the polynomials and storing in R all intermediate results of hashing. The verifier puts the polynomials together with intermediate hash values to generate $H(U)$ where U means collection of all polynomials.

The advantage of hierarchical hashing is evident when we consider the storage required to allow authentication of k public values using the following approach. An entity A authenticates t public values $Y_1, Y_2, \ldots, Y_t$ by registering each with a read-only registry or trusted third party. This approach requires registration of t public values, which may raise storage issues at the registry when t is large. In contrast, a Hierarchical Hashing requires only a single value registered in the registry. If a public key Y_i of an entity A is the value corresponding to a leaf in an authentication tree, and A wishes to provide B with information allowing B to verify the authenticity of Y_i , then A must (store and) provide to B both Y_i

and all hash values associated with the authentication path from Y_i to the root; in addition, B must have prior knowledge and trust in the authenticity of the root value R. These values collectively guarantee authenticity, analogous to the signature on the public-key certificate. The number of values each party must store is log(t).

Consider the length (or the number of edges in) the path from each leaf to the root in a binary tree. The length of the longest such path is minimized when the tree is balanced. i.e., when the tree is constructed such that all such paths differ in length by at most one. The length of the path from leaf to the root in a balanced binary tree containing t leaves is about log(t).

Using a balanced binary tree as authentication tree, with t public values as leaves, authenticating a public value therein may be achieved by hashing log(t) values along the path to the root.

Authentication trees require only a single value which is the root value, in a tree to be registered as authentic, but verification of the authenticity of any particular leaf value requires access to and hashing all values along the authentication path from leaf to root.

To change a public (leaf) value or add more values to an authentication tree requires re-computation of the label on the root vertex. For large balanced tree, this may involve a substantial computation. In all cases, re-establishing trust of all users in this new root value is necessary.

The computational cost involved in adding more values to a tree may motivate constructing the new tree as an unbalanced tree with the new leaf value being the right child of the root, and the old tree, the left. Another motivation for allowing unbalanced trees arises when some leaf values are referred far more frequently than others.

3.3 Security Issues

There follow some remarks on the security of the schemes:

- the scheme signs simultaneously all datagrams using a single signature. The important difference of this scheme from other schemes is that verification of datagrams can be done independently (or in parallel). In other words, to authenticate datagrams, one does not need to know all datagrams,

- no authentication is required for the coefficients fetched from read-only registry. This is because, if entries are tampered with, then packets will be rejected since the final hash recovered from the signature will be different from the hash value obtained from the datagrams and the polynomial,

- the only security problem could be of denial of service when an attacker may intentionally modify polynomial coefficients to reject the datagrams,
- in both flat and hierarchical k-sibling approaches, a single signature is required:
    1 the description of public polynomial coefficients used in the k-sibling intractable hashing takes about n integers each of size 160 bits for SHA1, where n is the number of packets of the message M and $k > 2$. If k=2 then this number $=2n$.

    2 the scheme may used against denial-of-service attacks. In particular, it would be able for receivers' at the intermediate nodes to ignore those packets that have failed to pass k-sibling hashing verification. attack from malicious source.
- the authentication scheme described above could be used for both types of IP data transfer modes: connection-oriented and connection-less. In the case of connection-oriented communication, where a node or destination sees almost all the packets of the message, flat sibling hash with a single polynomial $U(x)$ of degree n is best applicable. If, however, a node may see only a small fraction of packets, as in connection-less communication, then the hierarchical sibling with 2-sibling hashing seems to be superior.

4. ONE-TIME SIGNATURES

One-time signatures derived their importance from their fast signature verification, in contrast to typical digital signature schemes, which have either high generation or verification computation time. One-time signatures are a perfect option for authenticating particular types of applications were receivers are of low power capability, such as smart cards, or for online applications, such as video/audio streaming, which requires fast verification.

One-time signatures have to be efficient and secure. Typically, the verification of the signature is expected to be very efficient. Additionally, signatures have to be initialized well ahead of time when messages are to be signed and verified. This allows the signer to pre-compute the signature parameters so they can be fetched by potential verifiers. Once the message is known, the signer can sign it quickly and receivers can verify the signed message efficiently. A distinct characteristic of one-time signatures is that they are used once only. To sign a new message, the signer must initialize the signature parameters (parameters of old signatures must not be reused). The security of one-time signatures is measured by the difficulty of forging signature by an adversary who normally has access to a single pair: message and its signature.

Rabin [11], Merkle [10] and GMR [13] are well known examples of one-time signature schemes. Although they differ in their approaches, but they share the same idea: only one message can be signed using the same key. Once the signature is released, its private key is not used again, otherwise, it is possible for an adversary to compute the key.

One of the new approaches in designing such signatures is BiBa one-time signature [5]. BiBa is an acronym for BIns and BAlls. It uses bins and balls analogy to create a signature. To sign a message m, the signer first uses random precomputed values generated in a way that a receiver can authenticate them with a public key. These precomputed values are called SEALS. (SElf Authenticating vaLues). The signer then compute the hash of the message $h = H(m))$ and then compute the hash function G_h. Now, the collision of SEALS under a hash function G_h forms a signature: $G_h(s_i)$=$G_h(s_j)$, where $s_i \neq s_j$. The BiBa signature exploits the birthday paradox property, in that the signer who has a large number of balls finds a collision (signature) with high probability, but a forger who only has a small number of balls has a negligible probability of finding a signature.

The BiBa signature scheme has desirable features such as small authentication space overhead and fast verification time. However, its public keys are very large, and the time needed to generate a signature is higher than any other known system, and it requires parallel processors to find collision of SEALS. This makes signature generation a computation overhead. Also, it uses an *ad-hoc* approach to find collisions among the 'SEALS' to the corresponding bin which results the high signature generation time.

4.1 *K*-Sibling One-time Signature

We propose a variant approach of BiBa by using the SIFF method. SIFF provides hashing with a controlled number of easy-to-find collisions. In other words, we apply a *deterministic* approach in finding a collision (signature).

As for signatures based on public-key cryptography, we assume that we are going to produce signatures for digests of messages. Thus suppose that messages to be signed are of constant length (160 bits if we use SHA1).

Let $\text{SIFF}_i(x)$ be an instance of k-sibling hash function that for k inputs $x_{i,0}, \ldots, x_{i,k-1}$ produces the output α_i or

$$\text{SIFF}_i(x_{i,j}) = \alpha_i \text{ for } j = 0, \ldots, k-1$$

The function applies a polynomial $U_i(x) = u_{i,0} + u_{i,1}x + \ldots + u_{i,k-1}x^{k-1}$ that collides for the inputs $x_{i,0}, \ldots, x_{i,k-1}$ or

$$U_i(H(x_j)) = \alpha_i$$

where H is a collision-resistant hash function. Assume that the message to be signed is $M = (m_1, \ldots, m_t)$ where m_i are v-bit sequences. The message M consists vt bits (typically of the length 160 bits).

To design our one-time signature we use the sequence of t instances of SIFF where each instance applies 2^v collisions.

Initialization
The signer builds up the sequence of SIFF$_i(x)$ for $i = 1, \ldots, t$. He starts from SIFF$_1(x)$. First he picks up at random a sequence of 2^v integers (whose length is determined by the security parameter of the signature). Let the sequence be $r_{1,j}$; $j = 0, \ldots, 2^v - 1$ and denote $x_{1,j} = (r_{1,j}, j)$. The signer chooses at random the output α_1 and calculates the polynomial $U_1(x)$ such that

$$U_1(H(x_{1,j})) = \alpha_1 \text{ for } j = 1, \ldots, 2^v$$

Next, he creates SIFF$_i(x)$ for $i = 2, \ldots, t$. For each i, he selects at random integers $(r_{i,j})$; $j = 0, \ldots, 2^v - 1$, composes

$$x_{i,j} = (r_{i,j}, j, \alpha_{i-1})$$

and calculates the polynomial $U_i(x)$ such that

$$U_i(H(x_{i,j})) = \alpha_i$$

for a random α_i. The polynomials $U_i(x)$ and the final value α_t are made public in the read-only authenticated registry; $i = 1, \ldots, t$.

Signing

Given a message $M = (m_1, \ldots, m_t)$. The signer marks the input x_{1,m_1} and extracts r_{1,m_1} and similarly determines r_{i,m_i} for $i = 2, \ldots, t$. The signature is

$$S(M) = (r_{1,m_1}, \ldots, r_{t,m_t})$$

The pair $(M, S(M))$ is the signed message.

Verification

The verifier takes the pair $(\tilde{M}, S(\tilde{M}))$ and the public information, i.e. coefficients of polynomials $U_i(x)$ and α_t. Knowing $\tilde{x}_{1,\tilde{m}_1} = (\tilde{r}_{1,\tilde{m}_1}, \tilde{m}_1)$ and the polynomial $U_1(x)$, he can compute $\tilde{\alpha}_1$. Next, he recreates the inputs $\tilde{x}_{i,\tilde{m}_i} = (\tilde{r}_{i,\tilde{m}_i}, \tilde{m}_i, \tilde{\alpha}_{i-1})$ for $i = 2, \ldots, t$. If the last recovered $\tilde{\alpha}_t$ is equal to α_t recovered from registry, the signature is considered valid. Otherwise, it is rejected.

Suppose that an adversary knows a signed message and tries to modify either message or signature such that the forged (and signed) message passes verification. Obviously, the adversary also knows the public information. Informally, if the adversary is successful it means he was able to create either a new collision (which was not designed by the signer) or was able to guess one of the strings $r_{i,m'}$. The first event is excluded if we assume that the SIFF is collision resistant. The probability of the second event can be made as small as requested by choosing an appropriate length of the strings $r_{i,j}$. It is important to note that the above considerations are valid only if the public information about signatures is authentic.

The signature allows to trade efficiency of verification with the workload necessary to set up the signature system. This is very important aspect of the signature. Note, however, that the setup is done for each single message (this is one-time signature). Verification is done many times, typically as many times as there are different recipients. Consider two extreme cases: the first with the longest signature where SIFFs are designed for binary messages or $v = 1$ (t is the largest) and the second with $t = 1$. The first case permits a very efficient setup of the system with relatively small public information. The price to pay is bandwidth necessary to transport a very long signature and verification consumes a largest number of hash operations (as many as bits in the signed message).

The second case applies a relative small number of SIFFs (t is small). The setup is very expensive as any single SIFF applies large number of collisions and in effect the corresponding polynomials are very long. Receivers must fetch polynomial coefficients for verification. Verification seems to be fast as it requires a small number of hash operations. Signatures are relatively short.

Some scope for more efficient implementation exists if the strings $r_{i,j}$ are generated differently. Note that the system applies $t2^v$ such strings but only t are used as the signature. To reduce the necessary storage for keeping the strings by the signer, it is reasonable to choose at random $t+1$ integers $r_{i,1}$; $i = 1, \dots, t$ and the integer R. A polynomial $G(x)$ of degree t can be design such that $G(0) = R$ and $G(i) = r_{i,1}$ for $i = 1, \dots, t$. Note that other $r_{i,j}$ can be derived from the polynomial $G(x)$. This way of generation of $r_{i,j}$ is secure in the sense that signatures reveal t points on $G(x)$ leaving a single point on $G(x)$ unknown to the adversary.

5. CONCLUSION

The k-sibling intractable hashing function is a useful crytographic tool that might be used to solve a number of problems. We have exploited it to design a new authentication scheme that can verify the authenticity of independent messages. For example, it enables, intermediate nodes in a communication network to authenticate the source of packets. We have also used it to design a new one-time digital signature which has low computation and space overhead.

Appendix: A. Lagrange Interpolation Polynomial

The Lagrange interpolating polynomial is a polynomial of degree $n-1$ which passes through the n points $y_1 = f(x_1)$, $y_2 = f(x_2)$, ..., $y_n = f(x_n)$. It is given by:

$$P(x) = \sum_{j=1}^{n} P_j(x)$$

where,

$$P_j(x) = \prod_{k=1}^{n} \frac{x - x_k}{x_j - x_k} y_j$$

Written explicitly,

$$\begin{aligned} P_j(x) &= \frac{(x-x_2)(x-x_2)\dots(x-x_n)}{(x_1-x_2)(x_1-x_3)\dots(x_1-x_n)} y_1 \\ &+ \frac{(x-x_1)(x-x_3)\dots(x-x_n)}{(x_2-x_1)(x_2-x_3)\dots(x_2-x_n)} y_2 \end{aligned}$$

$$\vdots$$

$$+ \frac{(x-x_1)(x-x_2)\dots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\dots(x_n-x_{n-1})}y_n$$

References

[1] R. Gennaro and P. Rohatchi, " How to Sign Digital Streams", *Advances in Cryptology - CRYPTO'97*, Lecture Notes in Computer Science 1249, Springer-Verlag, pp. 180-197, 1997.

[2] R. Canetti, J. Garay, G. Itkins, D. Micciancio, M. Naor, B. Pinkas, " Multicast Security: A Taxonomy and some efficient Constructions", IEEE INFOCOM'99.

[3] A. Perrig, R. Canetti, J.D. Tygar, D. Song, " Efficient Authentication and Signing of Multicast Streams over Lossy Channels", ACM CCS'02 , 2000.

[4] M. Al-Ibrahim and J. Pieprzyk, "Authenticating Multicast Streams in Lossy Channels Using Threshold Techniques," in *Networking – ICN 2001, First International Conference,* Colmar, France, *Lecture Notes in Computer Science,* vol. 2094, P. Lorenz (ed), pp. 239–249, 2001.

[5] A. Perrig. "The BiBa One-Time Signature and Broadcast Authentication Protocol". ACM, CCS'01, Philadelphia, November 2001.

[6] M. Bellare, A. Desai, D. Pointcheval and Rogaway. "Relations among notions of security for public-key encryption schemes". In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO'98,* Lecture Notes in Computer Science No. 1462, Springer-Verlag, pages 26-45, 1998.

[7] Ivan Damgard. "A design principle for hash functions." In G. Brassard, Editor, *Advances in Cryptology - CRYPTO'89,* Lecture Notes in Computer Science No. 435. pages 416-427. Springer-Verlag, 1989.

[8] Y. Zheng, T. Hardjono and J. Pieprzyk. "The sibling intractable function family(SIFF): notion, construction and applications. *IEICE Trans. Fundamentals,* vol. E76-A:4-13, January 1993.

[9] J. Pieprzyk and E. Okamato. " Verifiable secret sharing", *ICISC'99,* Seoul, Korea, pages 169-183, Lecture Notes in Computer Science No. 1787. Springer-Verlag 1999.

[10] R. Merkle. " A Certified Digital Signature", *Advances in Cryptology - CRYPTO'89,* pages 218-238 Springer-Verlag, 1989. Lecture Notes in Computer Science No. 435.

[11] A. Menezes, P. Oorschot and S. Vanstone. "Handbook of Applied Cryptography", CRC Press, 1996.

[12] S. Kent and R. Atkinson. RFC 2401: Security architecture for the Internet Security. Networking Group, IETF, November 1998. http://www.ietf.org.

[13] S. Goldwasser, S. Micali, and C. Rackoff, " A Digital signature Scheme Secure Aagainst Adaptive Chosen-message Attacks", SIAM Journal on Computing, 17 (1988).

[14] P. Rohatchi. " A compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication", in *Proc. of 6th ACM conference on Computer and Communications Security,* 1999.

[15] M. Bellare, P. Rogaway. " Collsion-Resistant Hashing: Towards Making uniersal one way hash functions practical". *Advances in Cryptology - CRYPTO'97,*

Lecture Notes in Computer Science No. 1294, pages 470-484, Springer-Verlag, 1997.

[16] A. Shamir, "How to share a secret", *Communications of the ACM*, 22:612-613, November 1979.

[17] Oded Goldreich, Shafi Goldwasser, and Silvio Micali, " How to construct random functions", *Journal of the ACM*, 33(4):792-807, October 1986.

[18] S. Even, O Goldreich, S. Micali. "On-line/Off-line digital signatures", *Journal of Cryptology*, volume 9, number 1, winter 1996.