A NEW PUBLIC-KEY CRYPTOSYSTEM AND ITS APPLICATIONS

Akito Kiriyama, Yuji Nakagawa
School of Informatics, Kansai University, Takatsuki, Osaka, Japan

Tadao Takaoka, Zhiqi Tu
Department of Computer Science and Software Engineering, University of Canterbury
Christchurch, New Zealand

Keywords: Public-key cryptosystems, Non-linear knapsack problem, Access control.

Abstract: We propose in this paper a new public-key crypto-system, called the non-linear knapsack cryptosystem. The security of this system is based on the NP-completeness of the non-linear knapsack problem. We extend the system into secret sharing and access control. That is, an encrypted message can be decrypted only when all members of a group agree to do so with their secret sub-keys. The secret sharing here is equivalent to access control, which establishes multiple identities. That is, when the verifier challenges the prover with encrypted messages with public sub-keys, the prover can prove multiple identities using the secret sub-keys. Some experimental results are given, which demonstrate the efficiency of our system.

1 INTRODUCTION

In the era of ubiquitous computing, the technology of RFID (radio frequency identification) is receiving much attention. Viewing from another perspective, RFID brings up a tagged society, as described in Thinking Tags (Borovoy, 1996). Tags can be attached to almost everything; from manufacturing to retail, from human to animals. The attached tag emits a radio signal passively or actively. The tags can control manufacturing process up stream, and retail process down stream with various pieces of information recorded into the tags, which are attached to goods and commodities. A typical example of a tag attached to a human is seen in museums. A visitor to a museum attaches a tag to the chest that records his/her characteristics, such as gender, age, mother tongue, etc. Also payment information for special exhibition rooms is important for access control. This tag for a museum illustrates the importance of data security. General discussions of RFID are given in the recent issue of the Communications of the ACM, especially the example of museum (Hsi, et. al., 2005), data security (Ohkubo, et. al., 2005), and access control (Basker, et. al., 2005).

In this paper we develop a public-key cryptosystem suitable for RFID applications. Important features of the tag are

(1) Processing speed
(2) Its size, i.e., memory requirement
(3) Privacy and authenticity

While it is hard to achieve all of these features to a great satisfaction, we achieve (1) and (3) by our proposed cryptosystem while (2) is kept at a reasonable level.

Since the concept of public-key crypto-systems was published (Diffie and Hellman, 1976), there have been several concrete systems. They are classified into three categories. The first is based on the difficulty of factoring the product of two large prime numbers. The most famous is the RSA system invented by Rivest, Shamir and Adelman (Rivest, et. al, 1978). The second is based on the difficulty of discrete logarithm computation. A typical system here is the cryptosystem by (ElGamal, 1985). The difficulties of factoring and discrete logarithm are necessary conditions for the security of those systems; they have never been proven to be sufficient conditions, that is, those systems might be broken without factoring or discrete logarithm. The third is the knapsack cryptosystem suggested by Merkle and Hellman (Merkle and Hellman, 1978). This system is based on the NP-completeness of the linear knapsack problem. The systems in the first

Kiriyama A., Nakagawa Y., Takaoka T. and Tu Z. (2006).
A NEW PUBLIC-KEY CRYPTOSYSTEM AND ITS APPLICATIONS.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - ISAS*, pages 524-529
DOI: 10.5220/0002451105240529

two categories survived crypto attacks, and said to be safe for practical use. After the knapsack cryptosystem was first published, it was broken by Shamir's algorithm (Shamir, 1982), and later by the LLL algorithm (Lenstra, et. al., 1982), and (Lagarias and Odlyzko, 1985). To save the knapsack system, there have been many modifications for this system, such as (Adelman, 1983) and (Chor and Rivest, 1985). Unfortunately most of them have been broken by the above mentioned methods.

In this paper we propose a public-key cryptosystem based on the non-linear knapsack problem, a general form of which is known to be NP-complete. Although factoring and discrete logarithm are believed to be difficult, we have more evidence that NP-complete problems are difficult, which may boost the security of our proposed system. Also the non-linear property of the system will resist crypto attacks on the linear knapsack cryptosystems. However we have not proven that breaking our cryptosystem is NP-complete. Although our system is a general purpose one, we focus on its use for authentication, especially as application to access control. The authentication method to which we compare our system is the batching version by (Genaro et. al., 2004) of Schnorr's authentication scheme (Schnorr, 1991).

In terms of computing time, ignoring the alphabet size, our system can encrypt and decrypt the given message in $O(n^2)$ time where n is the size of the message. The RSA system and Schnorr's authentication system require $O(n^3)$ time to deal with n-bit integers if we use the standard $O(n^2)$ algorithm for multiplication and division of n-bit integers.

2 LINEAR KNAPSACK CRYPTOSYSTEM

Let $a_1,\ldots,a_n$ be n positive integers and $\mathbf{x}$ be a binary vector $\mathbf{x}=x_1x_2\ldots x_n$. Given an integer C, compute $\mathbf{x}$ that satisfies

$$C = \sum_{i=1}^{n} a_i x_i \qquad (1)$$

If x_i=(0) 1, we see a_i is (not) chosen to make C. This is like we pack some of items i whose size is a_i into a knapsack of size C. This problem is known to be NP-complete. Note that a_i are large integers given by n bits. Sequence $\{a_i\}$ is said to be super increasing if

$$\sum_{j=1}^{i-1} a_j < a_i \text{ for } i=1, \ldots, n$$

A super increasing sequence $\{a_i\}$ is chosen for a secret key. If $\{a_i\}$ is super increasing, the knapsack problem becomes easy and can be solved in $O(n^2)$ time as follows: We examine a_i for i=n,n-1,…,1. If a_n>C, we cannot choose a_n. Otherwise we need to choose a_n and decrease C by a_n, since we cannot make C with $a_1,\ldots,a_{n-1}$. The same reasoning proceeds for n-1,…,1 with C decreased as necessary.

The time of this algorithm is $O(n^2)$ as C:=C-a_i is executed at most n times, and each takes O(n) time. A multiplier w and a prime number p are chosen as part of secret key such that gcd(w, p)=1. The sequence $\{a_i\}$ is converted to $\{a_i'\}$ by $a_i'=a_iw \bmod p$.

After this conversion, sequence $\{a_i'\}$ is no longer super increasing, looking like random. This sequence $\{a_i'\}$ is published as the public key. The sender encrypts his binary message $\mathbf{x}=x_1x_2\ldots x_n$ as follows: Compute

$$C' = \sum_{i=1}^{n} a_i' x_i \qquad (2)$$

This knapsack problem seems difficult. After receiving C', the receiver computes $C=C'w^{-1} \bmod p$ and solve (1) for $\mathbf{x}=x_1x_2\ldots x_n$ by the decryption algorithm.

3 NON-LINEAR KNAPSACK CRYPTOSYSTEM

To overcome the weakness of the linear knapsack cryptosystem, we propose a non-linear cryptosystem in this section. Let $f_i(x)$ be a non-linear function of x, and $\mathbf{x}$ be an m-ary vector $\mathbf{x}=x_1x_2\ldots x_n$, that is, each x_i is regarded as a symbol of the alphabet {1,2,…,m}. Given an integer C, compute $\mathbf{x}$ that satisfies

$$C = \sum_{i=1}^{n} f_i(x_i) \qquad (3)$$

This problem is known to be NP-complete. In the linear knapsack problem, our decision was a binary one like whether to choose item i. In the present non-linear one, each item has m kinds such that the size of the j-th kind of item i is $f_i(j)$. The problem is to make C by choosing appropriate kinds of items 1,…,n.

We first make an easy knapsack problem, by creating integer values $f_i(x)$ from bit patterns. We prepare mask patterns using the following parameters:

n : number of items
m : number of kinds of each item
l : number of mask bits for each item.

The block size for encryption corresponds to n, and m plays the role of alphabet size for the plain text. The mask pattern of item i, mask(i), has *l*n bits of which *l* bits are 1 and the rest are 0, and satisfies the following condition. Let "&" be the bit-wise "and" operation in the following.

mask(i) & mask(j) = (000 … 0), all 0 for i≠j.

bitwise union of all mask(i), i=1,…,n, is (111…1), all 1.

Let value(i, j) be a vector not greater than mask(i) as a binary vector, where order **x**<=**y** for **x**=$x_1…x_n$ and **y**=$y_1…y_n$ is defined by x_i<=y_i for i=1,..,n. This mapping from j to value(i, j) is denoted by $f_i(j)$. We use the same notation for its numerical value. That is, $f_i(j)$ = value(i, j).

Lemma. mask(i) & $f_i(j) = f_i(j)$.

mask(i) & $f_k(j)$ = (00…0), all 0, for k≠i.

Example. n=4, m=3, *l*=2. There are four items (1, 2, 3, 4).

mask(1) = (01001000), mask(2) = (10010000)
mask(3) = (00100001), mask(4) = (00000110)

item {kind} --->{value(i, j) in binary sequence}
{numerical value in decimal}

---

1 {1, 2, 3}--->{00001000, 01001000, 01000000}
{8, 72, 64}
2 {1, 2, 3}--->{10010000, 10000000, 00010000}
{144,128, 16}
3 {1, 2, 3}--->{00000001, 00100000, 00100001}
{1, 32, 33}
4 {1, 2, 3}--->{00000100, 00000110, 00000010}
{4, 6, 2}

A secret key is chosen as mn integers as follows:

A = ($f_1(1)$, $f_1(2)$, …, $f_1(m)$)
($f_2(1)$, $f_2(2)$, …, $f_2(m)$)
…
($f_n(1)$, $f_n(2)$, …, $f_n(m)$)

The knapsack problem with this definition of $f_i(j)$ is easy, since the kind of each item can be sieved by the mask of the item.

Example. A = (8, 72, 64)
(144, 128, 16)
(1, 32, 33)
(4, 6, 2)

Let $p > 2^{ln}$ be a prime number, w be such that gcd(w, p)=1, and w^{-1} be the multiplicative inverse of w mod p. Let B be obtained by operating "w times mod p" on each component of A. We express this by B = Aw mod p, and thus A = Bw^{-1} mod p.

Example. p=283, w=200, w^{-1} mod p = 75.

B = (185, 250, 65)
(217, 130, 87)
(200, 174, 91)
(234, 68, 117)

The correspondence from kinds to values in B is denoted by $f'_i(j)$=value'(i, j). That is, $f'_i(j)=f_i(j)w$ mod p and $f_i(j)=f'_i(j)w^{-1}$ mod p.

Example. $f'_1(1)$ = value'(1, 1) = 185, etc.

Encryption. Let **x**=$x_1x_2…x_n$ be an m-ary sequence of length n to be encrypted. The cryptogram C is computed by

$$C = \sum_{i=1}^{n} f'_i(x_i)$$

Decryption. Compute $M=Cw^{-1}$ mod p

Compute $y_i=f_i^{-1}$(mask(i)&M) for i=1,…,n.

Let **y**=(y_1, …, y_n) be the decrypted message. It is straightforward to prove **y** = **x**. The inverse function f_i^{-1} is implemented by a hash table.

Example. **x** = 1 2 3 1. This message is encrypted as follows:

C'=f_1'(1)+f_2'(2)+f_3'(3)+f_4'(1)
=185+130+91+234 = 640
$C = C'w^{-1}$ mod p = 640*75 mod 283 = 173
Binary expansion of 173 = 10101101

173&72 = 8 ➔ kind 1, 173&144 = 128 ➔ kind 2
173&33 = 33 ➔ kind 3, 173&6 = 4 ➔ kind 1

We note that the bit pattern in each mask must be random and that it will be safe not to use all binary vectors covered by the masks, as small values such as 1 will expose the multiplier w. More security considerations will be given in Section 5.

4 ENCRYPTED SECRET SHARING

(Encrypted) secret sharing is the property of a group that only when the members of the group agree, they can (decrypt) read some messages.

Let a message X be embedded in a vector **x** = (X, R_1,…,R_{n-1}), where R_i are random messages. We assume all are integers less than a large prime p, and all operations are done with "mod p". If we multiply this vector with a regular (n, n) matrix S to compute **c**=**x**S, where **c**=(C_1,…,C_n), we can distribute the secrecy of X into n pieces of information **c**. By putting them together, and computing **x**=**c**S^{-1}, we can read X. If any C_i is missing, we can not read X. In this framework, there is no encryption; if **c** and S are intercepted, X can be read by outsiders. Normally shares are not encrypted. See (Shamir, 1979).

We combine our non-linear knapsack cryptosystem and the above mentioned secret sharing in this section. We first describe our method for two members A_1 and A_2 in the group. Let $f^1_i(j)$ and $f^2_i(j)$ be defined by

$f^1_i(j) = f_i(j)w_1 \bmod p$, $f^2_i(j) = f_i(j)w_2 \bmod p$

$f^1_i(j)$ and $f^2_i(j)$ are the public keys for A_1 and A_2. $\{f_i(j)\}$ corresponds to the master key and w_1 and w_2 correspond to sub keys. The sender can use these public keys, regarding A_1 or A_2 as single users. A_1 and A_2 can decrypt their incoming messages using w_1 and w_2, and $\{f_i(j)\}$. Note that $\{f_i(j)\}$, w_1, w_2 , and p are co-owned within the group. Suppose the sender wants to send an encrypted message for the group members to read at the same time. The sender chooses a random number (or random message) R. Let message $\mathbf{x}=x_1x_2\ldots x_n$ be encrypted into two cryptograms C_1 and C_2 by

$$C_1 = \sum_{i=1}^{n} f^1_i(x_i) + R \equiv \left(\sum_{i=1}^{n} f_i(x_i)\right)w_1 + R) \bmod p$$

$$C_2 = \sum_{i=1}^{n} f^2_i(x_i) + R \equiv \left(\sum_{i=1}^{n} f_i(x_i)\right)w_2 + R) \bmod p$$

In the above, the right-hand side of "≡" is given only for explanation purpose; the sender does not know p, and the "mod p" operation is done at the receiver side. From this simultaneous linear equation, the receivers compute

$$M = \sum_{i=1}^{n} f_i(x_i) = (C_1 - C_2)(w_1 - w_2)^{-1} \bmod p$$

This can be done only when w_1 and w_2 are known. After this, using the previous decryption procedure, the receivers, namely the group members, recover $\mathbf{x} = x_1x_2\ldots x_n$.

Example. $w_2=190$, $p=283$, $w_2^{-1}=213$, $R=100$

$\{f_i(j)\} = (\ 8,\ 72,\ 64)$
$(144, 128,\ 16)$
$(\ 1,\ 32,\ 33)$
$(\ 4,\ 6,\ 2)$

$\{f^1_i(j)\} = (185, 250,\ 65)$
$(217, 130,\ 87)$
$(200, 174,\ 91)$
$(234,\ 68, 117)$

$\{f^2_i(j)\} = (105,\ 96,\ 274)$
$(192, 265, 210)$
$(190,\ 137,\ 44)$
$(194,\ 8,\ 97)$

Let $\mathbf{x} = (1, 2, 3, 1)$

$C_1 = 185 + 130 + 91 + 234 + 100 = 640 + 100$
$= 740 = 200M + R \bmod 283$

$C_2 = 105 + 265 + 44 + 194 + 100 = 608 + 100$
$= 708 = 190M + R \bmod 283$

$32 = 10M \bmod 283$, $M = 32*10^{-1} \bmod 283$
$= 32*85 \bmod 283 = 173$

This two-member secret sharing system can be generalized into k members in the following. We go with k=3 for illustration. Let us have one more multiplier w_3, and public key $f^3_i(x)$, derived from $f_i(j)$. After choosing a plain text $\mathbf{x}=x_1x_2\ldots x_n$, the sender computes

$$C_1=\sum_{i=1}^{n} f^1_i(x_i) +R_1 + R_2 \equiv(\Sigma f_i(x_i)w_1+R_1+R_2)\bmod p$$

$$C_2=\sum_{i=1}^{n} f^2_i(x_i)+R_1+2R_2 \equiv(\Sigma f_i(x_i)w_2+R_1+2R_2)\bmod p$$

$$C_3=\sum_{i=1}^{n} f^3_i(x_i) +R_1+3R_2 \equiv(\Sigma f_i(x_i)w_3+R_1+3R_2)\bmod p$$

From this the receivers compute

$C_1 -2C_2 + C_1 \equiv (w_1 -2w_2 + w_3)M \bmod p$

$M \equiv (C_1 -2C_2 + C_3)(w_1 -2w_2 + w_3)^{-1} \bmod p$

Using the matrix/vector notation and omitting "mod p", we have

$C_1 = Mw_1 + R_1 + R_2$
$C_2 = Mw_2 + R_1 + 2R_2$
$C_3 = Mw_3 + R_1 + 3R_2$

$$(C_1, C_2, C_3) = (M, R_1, R_2) \begin{vmatrix} w_1 & w_2 & w_3 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{vmatrix}$$

We call this matrix the verification matrix and denote it by V. Vertical bars are used for expressing matrices. The verification matrix minus the first row, denoted by V*, is shared by the sender and the receivers.

$$V^* = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{vmatrix}$$

This V* is chosen just for illustration. As long as its rank is k-1, any matrix will do.

Plain text $\mathbf{x}$, and random messages R_1 and R_2 are generated by the sender. In V, we have $w_1,\ldots,w_k$ in the first row, and some constant values for the rest. The verification matrix must be regular. If the rank of V* is k-1 and we generate $w_1,\ldots,w_k$ random, the probability that the matrix is singular is low. The users can check this in advance.

The inverse matrix V^{-1} is given by

$$\begin{vmatrix} 1 & 2w_3 - 3w_2 & w_2 - w_3 \\ -2 & 3w_1 - w_3 & w_3 - w_1 \\ 1 & w_2 - 2w_1 & w_1 - w_2 \end{vmatrix} (w_1 - 2w_2 + w_3)^{-1}$$

Let d_{ij} be the (i, j) cofactor of the verification matrix V. Then $V^{-1} = (d_{ij})^T|V|^{-1}$, where $|V|$ is the determinant of V. Thus the first column, $(I_1,\ldots,I_k)^T$, of the inverse matrix is given by

$(d_{11}, \ldots, d_{1k})^T|V|^{-1}$. ("T" for transposition)

If $(I_1,\ldots,I_k)$ is pre-computed, we can compute M by the inner product of $(C_1,\ldots,C_k)$ and $(I_1,\ldots,I_k)^T$, taking O(k) operations of multiple precision numbers. To compute M we do not need the whole inverse matrix.

The rank of the (k-1, k) sub-matrix, V*, must be k-1. Otherwise M can be computed with less than k equations.

Example. $C_1 = Mw_1 + R_1 + R_2$
$C_2 = Mw_2 + R_1 + R_2$
$C_3 = Mw_3 + R_1 + 3R_2$

We have $C_1 - C_2 = (w_1 - w_2)M$ and $M = (w_1 - w_2)^{-1}(C_1 - C_2)$. That is, we can compute M without the knowledge of C_3.

We can use this system for secret sharing with threshold. That is, if an encrypted message is sent to k receivers, and they can decrypt the message only when t members agree. We make the matrix V* of rank t-1. Then we can solve the simultaneous equation of size t, and get M.

Example. $C_1 = Mw_1 + R$
$C_2 = Mw_2 + R$
$C_3 = Mw_3 + R$

If any two of the three members agree, M can be solved.

We can use the above secret sharing for access control. A user is granted several access rights, by receiving $w_1, w_2, \ldots$. A typical application is entrance fees in a museum. If a customer pays fees for entering several exhibition rooms, those $w_1, w_2, \ldots$will be provided. The server at each door will challenge the visitor with some secret message to see if it is decrypted and sent back to the server.

5 ANALYSIS

We define the *size* of a number by the number of bits or digits to express it. To prevent crypto-attack by exhaustive search, we propose to have the size of n around 50 or larger. The size of mask is given by O(ln). The size of modulus p and multiplier w is of the same order.

The summation in (3) takes O(ln^2) time. The bitwise "and" operation for decryption takes O(ln) time each, resulting in O(ln^2) time in total. The multiplication and division take O(l^2n^2) time, which is dominant. For the choice of secret keys, we suggest to choose random l/2 bits for 1 and the rest for 0 for each $f_i(j)$. The number of kinds, m, cannot be large to avoid the equal-sum event described later in this section. In the experiment, we set m=l/2.

Possible attack on our system can be considered from two angles. The first is to compute the secret key from the public key. If we try all possible w^{-1} on B, and see if each row can represent some mask pattern, it will take at least O(2^{ln}) time, which is not practical. If l is small, say 2 or 3, we generate all possible secret keys in polynomial time, and by comparing those with public ones, we can guess p and w in polynomial time, whereby secret keys can be guessed.

The other attack would be to solve the difficult knapsack problem. Although our non-linear knapsack problem has low density solutions, the attacking methods such as the LLL method are based on the linearity of the knapsack problem, and not directly applicable to our system. The direct exhaustive search will cost O(m^n) time.

Now we discuss how to set up the secret key. If we use just a few of 2^l bit patterns except all 0 in the mask(i) for the kinds of item i, that is, we use them sparsely, we introduce a super-increasing property on some of those kinds, which may become vulnerable to Shamir's crypto-attack. For example, if there is only one 1 in the l-bit pattern for $f_i(j)$'s, they are super-increasing.

If we have l mask bits, it would be safe to use l/2 ones for the function $f_i(j)$ for each item i and kind j. If we use all such bit patterns, we invite crypto-attack in the following way. Suppose we have four values (a, b, c, d) for $f_i(j)$ such that

$$a+b=c+d \quad (*).$$

Let (a', b', c', d') be the corresponding public key values. It is straightforward to see

$$a'+b'-(c'+d')=kp \quad (**)$$

for some k. Let us call this situation where we have several secret key at the left-hand side and right-hand side of the above equation (*), the equal-sum event. If we find another set of such four public key values with k'p for the right-hand side of the above equation (**), the value of p can be revealed by the Euclidian algorithm, which will in turn break the whole system. To prevent this situation from happening, we suggest to choose random l/2 bits for 1 and the rest for 0 for each $f_i(j)$. The number of kinds, m, cannot be large to avoid the above situation.. This is because the number of the equal-sum events is given by

$$\sum_{i=0}^{m} \sum_{j=0}^{m-i} C(m, i)C(m-i, j) = 3^m,$$

where C(m, i) is the binomial co-efficient of i out of m. The probability of two l-bit sequences are equal is 2^{-l}, and the probability of one equal-sum event is bounded by this value. Thus the probability of the above situation is bounded by $(3^m 2^{-l})^2 = (2^{1.58m - l})^2$. For l=20 and m=10, this probability is about 1/300. We need to multiply this probability by n, the number of items. Then the probability is about ¼. For security we would need to exhaustively check the equal-sum event after the secret key is set up. In this parameter setting, we hit an equal-sum event once every four attempts, in which case we generate the secret again until we have no equal-sum event.

6 EXPERIMENTAL RESULTS

As we mentioned above, we can use the secret sharing scheme for access control with multiple identities authentication. The following are our experiments on timing results based on the method in Section 4, compared with the timing results on the batch Schnorr scheme (Genaro, et. al., 2004) with the same number of identities. As for the selections of security parameters, they chose the following setting:

Number of identities: d = 32
Prime number q: let q be 200 bits
Prime number p: let p be about 1500 bits
Microchip PIC16LF628 microcontroller
Runnig time: about 2 seconds on

The implementation of our scheme is in the C language on a Linux machine with a Celeron processor with 2.2 GHz. In order to treat two schemes under the same condition, our selections of security parameters are set as follows:

Number of identities: d = 32
Number of items: n = 75
Number of kinds: m = 10
Number of mask bits for each item: $l = 20$

Each mask pattern's length: $ln = 20*75 = 1500$

Prime number p: $p > 2^{ln}$. So $p > 2^{1500}$, p is of 451 decimal digits, which has a comparable size with the batch Schnorr scheme.

Those parameters are set to match the performance measurement of the batch Schnorr. For our experiment we did encryption/decryption of a file with about 50 characters as a challenge with the total time of 0.057 sec. The memory requirement for the above selections of security parameters is 0.134 MB.

7 CONCLUDING REMARKS

We presented a new crypto-system based on the non-linear knapsack problem. At the moment, there is no attacking method for this crypto-system. The computational complexity of our system is low, and thus very efficient in practice. Also the simple structure is suitable for hardware implementation, especially as the bitwise "and" operation can be executed by a logic array in parallel. Only one problem is that the key size is rather large, that is, $O(lmn^2)$, whereas that of RSA is O(n). As the cost of memory chip is going down, this disadvantage will not be a great obstacle to our system.

We extended our system for encrypted secret sharing. This became possible as our non-linear system is not exactly "non-linear", but "super-linear", in the sense that we have non-linear functions at the bottom, which are gathered by the linear operation of summation. We incorporated linear algebra into this linear part at the top. There can be more generalizations from various angles.

REFERENCES

Adelman, L., 1983. *On breaking generalized knapsack public-key cryptosystems*, Porc. ACM Symp. On Theory of Computing 1983, pp402-412

Borovoy, R et. al., 1996. *Things that blink: Computationally augmented name tags*, IBM System Journal, vol. 35, no. 3 & 4, pp488-493

Chor, B. and R. L. Rivest. A, 1985. *Knapsack type public key cryptosystem based on arithmetic finite fields*, IEEE Trans. on Information Theory, IT-34, 1988, pp901-909

Diffie, W. and M. Hellman, 1976. *New directions in cryptography*, IEEE Trans. on Information Theory, IT-22, 6, pp644-654

ElGamal, T., 1985. *A public key cryptosystem based on discrete logarithms*, IEEE Trans. On Information Theory, IT-31, 4, pp469-472

Gennaro, R., D. Leigh, R. Sundaram, and William, 2004. *Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices*, AsiaCrypt 04, LNCS 3329, pp276-292

Hsi, S and Fait, H., 2005. *RFID enhances visitors' museum experience at the exploratorium*, CACM vol. 48, no. 9, pp 60-65

Lagarias, J. C., A. M. Odlyzko, 1985. *Solving low density subset sum problems*, JACM, vol. 32, 229- 246

Lenstra, A. K., H. W. Lenstra, Jr. and L. Lovasz, 1982, *Factoring polynomials with rational coefficients*, Math. Ann. 261

Merkle, L. C. and M. E. Hellman, 1978. *Hiding information and signatures in trapdoor knapsacks*, IEEE Trans. on Inf. Theory, 24, pp525-530

Ohkubo, M., Suzuki, K., and Kinoshita, S., 2005. *RFID privacy issues and technical challenges*, CACM vol 48, no 9, pp 66-71

Raskar, R, Beardsley, P, Dietz, P, and van Baar, J, 2005. *Photosensing wireless tags for geometric procesures*, CACM vol 48, no. 9, pp 46-51

Rivest, R. L., A. Shamir and L. Adelman, 1978. *A method for obtaining digital signatures and public-key cryptosystems*, CACM, 21, 2, pp120-126

Schnorr, C. P., 1991. *Efficient signature generation by smart cards*, J. of Cryptology, 4, 3, pp161-174,

Shamir, A, 1979. *How to share a secret*, CACM vol. 22, no. 11, pp612-613

Shamir, A., 1982. *A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem,* FOCS 1982: 145-152