Oblivious Transfer with Verification

Subhash Kak

Abstract— **Although random sequences can be used to generate probability events, they come with the risk of cheating in an unsupervised situation. In such cases, the oblivious transfer protocol may be used and this paper presents a variation to the DH key-exchange to serve as this protocol. A method to verify the correctness of the procedure, without revealing the random numbers used by two or more parties, is also proposed.**

Index Terms— **Cryptography, network security, multiparty communication, piggyback protocol**

I. INTRODUCTION

The generation of events of specific probability is essential in many computations and in simulation of physical processes. Of particular interest is the generation of a random sequence that can simulate physical noise and be used for cryptographic and coding purposes. In a random binary (0, 1) random sequence, where the bits are independent, the probability of each new bit being 0 (or 1) is 1/2.

If two parties (Alice and Bob) wish to determine who should play first at a game, they might agree to let Alice play first if she calls the next bit (or the nth future bit) correctly. The problem with this method is that if the algorithm generating the random sequence is known to, say, Alice, she can run it in advance and, therefore, know the bit in advance. To thwart such a possibility, one would need to place constraints on the nature of the random number generator such as designing it in such a way that it is impossible to emulate it. But that is not a realistic assumption if the generator is an algorithm that is implemented on a computer. If it is easy to generate a pseudo-random sequence, most likely it is cryptographically weak [1]-[7].

Alternatively, one could imagine that a trusted third party has a collection of random number generators. Alice now has to call the i^{th} outcome of the k^{th} random number generator correctly in order to win the call. If the number of generators is large and the number i is derived from some step in a computationally hard number-theoretic problem (such as the number of prime partitions of a large even number), it will become well-nigh impossible for cheating to occur. This is equivalent to the method of puzzles for security [8].

Manuscript received February 2, 2017. This work was supported by National Science Foundation grant #1117068.

Subhash Kak is with the Oklahoma State University, Stillwater, OK, USA (phone: 405-744-6096; e-mail: subhash.kak@okstate.edu).

For those who seek mathematical elegance, one might appeal to quantum theory [9]. The outcome of a superposition quantum state, such as $a|0\rangle+b|1\rangle$ is random, with the probability of 0 and 1 being $|a|^2$ and $|b|^2$, respectively. All one needs to do is to start with the state

$$\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$$

, and measure it along the $|0\rangle$ and $|1\rangle$ bases, and the chosen outcome will have a probability of exactly 1/2. An example of this are diagonally polarized photons that will be unpredictably received as horizontally or vertically polarized photons along these measurement bases.

This approach via physics is the perfect way to generate random events but it is not easy to implement [10]-[12]. Due to the Heisenberg's Uncertainty Principle, one cannot generate single quantum states at specified time instants. Indeed, a low-power laser will generate photons with a Poisson distribution [13]. If there are multiple photons with diagonal polarization, the pattern of reduction to the bases states will make it difficult to fix event probabilities. The randomness of collapse is at the basis of quantum cryptography protocols [14][15]. But due to the difficulty of generating single photon states, quantum cryptography itself uses classical random number generators to guide polarization rotations.

Classical randomness is viewed as an aggregate of countless quantum processes. One could have a trusted party look at the thermal noise across a resister at specified future time (so that the bandwidth of the measurement apparatus can be discounted) and check if it is greater or less than the zero threshold. This can serve as an effective method of generating random events. But this requires a trusted third party to supervise the event generation process.

The other method to use is the oblivious transfer (OT) protocol [16][17], where two parties mutually arrive at the probability event. In the most basic form of OT, the sender sends a message to the receiver with probability 1/2, while remaining oblivious as to whether or not the receiver obtained the message. Other probabilities can also be likewise generated [18]. These schemes depend on one-way, number-theoretic functions that are at the basis of public key cryptography [19] and they require a choice out of two alternatives to be made at some point in the process.

We assume that the two parties are authenticated to each other and the owner of the secret is honest (the recipient has no reason not being so). To ensure there is no cheating, one could speak in general either of post-communication audit, or supervision of the process by a trusted third party. The audit or verification process should not reveal the random numbers used by the two parties since that could compromise the random number generators used and weaken the security of the process.

We mention parenthetically that randomness was an important notion in ancient societies. The gods were taken to act randomly in a fashion that could not be understood by reasoning. The idea of Vedic ritual [20], Dionysian mysteries, the ecstatic trance of the Oracle of Delphi [21],[22], or shamanic practices of other cultures [23] was to get into a state where one could somehow connect to the time of the gods. The oracle's prophecy was worded ambiguously and what meaning it might convey could not be known to the oracle.

Here we show that an adaptation of the DH key exchange protocol will serve as an OT protocol with verification. We show that the protocol allows Bob to guess Alice's secret with the specified probability. Since the secret belongs to Alice, one can visualize a situation where she cheats so as to reduce Bob's guessing probability. We address this possibility and show how there can be verification of the procedure.

II. THE PROTOCOL FOR TWO PARTIES

Alice and Bob together (or a trusted party) choose and publish a large prime p and two integers u_1 and u_2 of large order modulo p. It may thus be assumed that both parties know that $u_1 = k\, u_2$.

Step 1. Alice chooses a random integer a, picks one of the two integers u_1 and u_2 and computes $A = u_i^a \bmod p$, where $i = 1$ or 2, and sends it to Bob.

Step 2. Bob chooses a random integer b, picks one of the two integers u_1 and u_2 and computes $B = u_j^b \bmod p$, where $j = 1$ or 2, and sends it to Alice.

Step 3. Alice takes the received number B and computes $B^a \bmod p = u_j^{ab} \bmod p$ as the key to be used in encrypting a secret file to be sent to Bob.

Step 4. Bob takes the received number A and computes $A^b \bmod p = u_i^{ab} \bmod p$ as the key to be used in decrypting a secret file received from Alice.

This protocol is shown in Figure 1 for the special case where Alice and Bob have chosen u_1 and u_2, respectively. The other cases are where the choice is flipped or where both Alice and Bob choose the same basis.

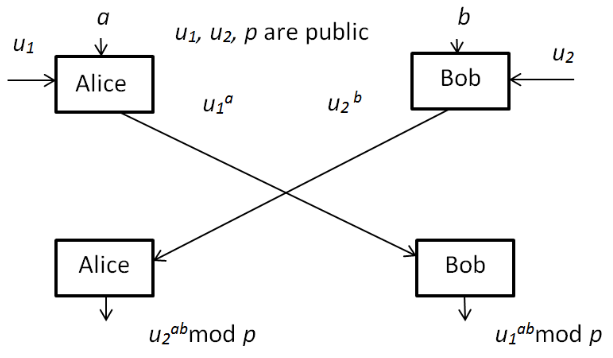

Figure 1. The proposed protocol where Alice and Bob choose different bases

It is assumed that Alice will use the key $u_2^{ab} \bmod p$ to code her secret. She does not know whether Bob possesses this key or $u_1^{ab} \bmod p$. The probability that they choose different bases is ½. Therefore, there is a 0.50 probability that the key generated by Alice and Bob is identical.

S
Alice
Bob
S

Figure 2. Bob gets the secret, S, if his key is the same as Alice's

If Bob fails to decrypt the secret with his key, he cannot use the knowledge that $u_1 = k\, u_2$, to determine the "correct" key. His incorrect key is related to the correct one through the relationship:

$$u_1^{ab} = u_2^{ab} k^{ab} \bmod p \tag{1}$$

Bob knows b, k, and $u_1^{ab} \bmod p$, but that is not sufficient to obtain the correct key unless he can solve the discrete logarithm problem.

The eavesdropper also cannot obtain any information about the final key from her observation of the data exchanged by Alice and Bob.

Generalization. If in the protocol, there are m bases, $u_1, u_2, \ldots, u_m$, rather than just two, as in the example above, the probability that Bob will know the secret is $1/m$.

III. Possible Cheating by Alice

Alice can cheat by not sending u_2^{ab} mod p to Bob over the public channel, but rather u_2^{fb} mod p, using the exponent f to build this fake key. This cheating will be evident if both Alice and Bob choose the same basis, which will happen 50% of the time. The case of cheating thus corresponds to the use of different exponents by the two parties.

To prevent cheating, we add the following steps to the protocol:

> *Step 5.* A random number r, publicly declared in advance, is used by Alice to generate $v^n = u_j^{abr}$ mod p ($n=abr$). In the example of Figure 1, $v^n = u_2^{abr}$ mod p. The number v^n is sent to Bob.
>
> *Step 6.* Bob uses the verification sequence $G(n) = v^n + w^n$ mod p to establish that there has been no cheating.

If $v = w$, $G(n) = 0$. When $v \neq w$, $G(n) = \alpha G(n-1) + \beta G(n-2)$ mod p, where α and β are constants that are easily found. The verification sequence G(n) is described in the next section.

If Alice were to cheat by using u_2^{fb} mod p as the key, but sends the correct u_2^n mod p, she will be exposed in case Bob has chosen u_2 and finds G(n) =0, while remaining unable to decrypt the secret.

IV. The Verification Sequence

Consider the sequence $G(n) = v^n + w^n$ mod p. In general we can write

$$v^k = \alpha_k v + \beta_k \bmod p$$
$$w^k = \alpha_k w + \beta_k \bmod p \qquad (2)$$

Theorem 1

$$G(n) = \alpha_k G(n-k+1) + \beta_k G(n-k) \bmod p \qquad (3)$$

Proof. $G(n) = (v^n + w^n) \bmod p$

$$= (v^{n-k}v^k + w^{n-k}w^k)$$
$$= v^{n-k}(\alpha_k v + \beta_k) + w^{n-k}(\alpha_k w + \beta_k)$$
$$= \alpha_k(v^{n-k+1} + w^{n-k+1}) + \beta_k(v^{n-k} + w^{n-k})$$
$$= \alpha_k G(n-k+1) + \beta_k G(n-k) \bmod p$$

When $k = 2$,

$$G(n) = \alpha\, G(n-1) + \beta\, G(n-2) \bmod p \qquad (4)$$

This means that the sum of successive powers of v and w suffices to establish that they have been computed to the same exponent. All that is required to find the values of α and β is the solution to equation (2) for $k = 2$. No knowledge of the actual value of n is needed while computing equation (4).

Example 1. Let k=2, v=3, and w=7 mod 19. To find α and β, we solve the equations:

$$3^2 = 9 = \alpha 3 + \beta \bmod 19$$
$$7^2 = 11 = \alpha 7 + \beta \bmod 19$$

We find that α=10 and β=17.

The series $G(n) = 3^n + 7^n$ mod 19, for n = 0, 1, 2, 3 … is as follows: 2, 10, 1, 9, 12, 7, 8 …
for which each nth element is $10\, G(n\text{-}1)+17\, G(n\text{-}2)$ mod 19.
For example, the value 9 is 10×1+17×10 mod 19.

Example 2. Let k=2, v=3, and w=5 mod 17. To find α and β, we solve the equations:

$$3^2 = 9 = \alpha 3 + \beta \bmod 17$$
$$5^2 = 8 = \alpha 5 + \beta \bmod 17$$

We find that α=8 and β=2.

The series $G(n) = 3^n + 5^n$ mod 17, for n=0, 1, 2, 3… is as follows: 2, 8, 0, 16, 9, 2, 0, 4, 15, 9 …
for which each *nth* element is $8\, G(n\text{-}1)+2\, G(n\text{-}2)$ mod 17. Theorem 1 may be extended to modulo m, if u and v are relative prime to m. If the exponents in equation (2) are not the same then the result of Theorem 1 will not be valid.

Since v and w are known, three consecutive $G(n)$ values can be computed by successive multiplication with the appropriate bases and it checked if the numbers have the relationship of equation (3).

V. Three or More Parties

Consider communicating parties Alice, Bob, and Charlie (the list can be augmented but here for simplicity we only speak of three) who wish to perform a secure computation, which is the sharing of random number. The first thing to be done is to create aliases so that actions within the computation are protected by the complexity of the computation. Each of these aliases is a random number. The three also wish to generate a single number that connects them with the multiparty computation.

In a centralized system (Figure 3), the trusted authority T performs the computation on the numbers *a, b, c* sent respectively by Alice, Bob, and Charlie. The numbers should be sent to T in a manner that hides each sender's identity. This requires a privacy preserving transformation where this hiding is accomplished by means of an appropriate one-way function.

Let the transformation carried out by T map the numbers to the range, R, which is [0, 1]:

$$R = T(a,b,c) \qquad (5)$$

R maps to different probabilities p_{ALICE}, p_{BOB}, $p_{CHARLIE}$ for the three communicating parties. This mapping may be done by assigning non-overlapping one-thirds of the range [0, 1] to the three parties.

$$p_{ALICE}, p_{BOB}, p_{CHARLIE} = f_i\,(R) \qquad (6)$$

The difficulty with this centralized procedure is that the users do not know if the transformation T is good at randomization. Although there is no way for them to confirm that the output R has a distribution which is uniform over [0, 1], a strong hashing function will be considered satisfactory in most cases. Centralized procedures are implemented in many computer-controlled applications like the ones in a casino or in online gambling. In these latter applications, the assignment of probabilities is determined by the nature of the computation (or game) and the house is also assigned a certain portion of the take in accordance with law.

In the decentralized system (Figure 3), after the users have been authenticated by some other protocol, they will send their random numbers *a, b,* and *c* to each other. This procedure is more than just a pairwise exchange of random numbers as in the standard DH protocol, since a product of the three must also be exchanged.

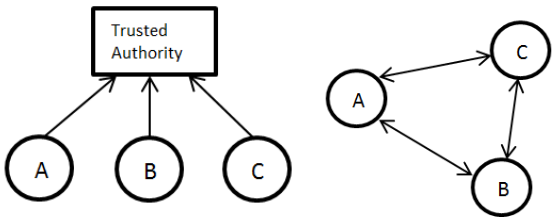

Figure 3. Centralized system with trusted authority; (right) decentralized system

In the case of four parties and two bases (*u* and *w*), the following cases will be different:

i. All chosen bases are the same (in which case the keys would be identical)
ii. Three choose one base and the fourth chooses another
iii. Two adjacent parties choose one base and the other two pick a different one
iv. Two non-adjacent parties choose one base and the other two pick the other

The cases ii, iii, and iv are described by Tables 1, 2, and 3, respectively.

Table 1.

A	B	C	D
u	u	u	w
w^{ad}	u^{ab}	u^{bc}	u^{cd}
u^{acd}	w^{abd}	u^{abc}	u^{bcd}
u^{abcd}	u^{abcd}	w^{abcd}	u^{abcd}

Table 2.

A	B	C	D
u	u	w	w
w^{ad}	u^{ab}	u^{bc}	w^{cd}
w^{acd}	w^{abd}	u^{abc}	u^{bcd}
u^{abcd}	w^{abcd}	w^{abcd}	u^{abcd}

Table 3.

A	B	C	D
u	w	u	w
w^{ad}	u^{ab}	w^{bc}	u^{cd}
u^{acd}	w^{abd}	u^{abc}	w^{bcd}
w^{abcd}	u^{abcd}	w^{abcd}	u^{abcd}

In case (ii), B and D share the key with A; in case (iii), only D shares the key with A; and in case (iv), C shares the key with A. Since the key generation process has three steps (represented by the three bottom rows of each table), the base travels one step to the right at each stage, ending up 3 positions to the right which is equivalent to one position to the left.

In Table 1, the total favorable probability of one of the three (B, C, D) obtaining the same key as A is 4/9 as shown in Table 4:

Table 4.

A	B	C	D	Result
u	u	u	w	A, B, and D share key
u	u	w	u	A, C, and D share key
u	w	u	u	B, C, and D don't share key with A
w	u	u	u	A, B, and C share key

If sharing of key with A by B, C, and D is represented by 1, these four cases represent the sequences 101, 010, 000, and 110. The cases of Table 4 map to the sequences 001, 100, 011, and that of Table 5 to the sequence 010.

Clearly, such analysis can be extended to more general cases. The protocol for three parties begins with a pairwise exchange of random numbers and then the product of the three:

Step 1. Alice and Bob share u^{ab} mod *p*, Bob and Charlie share u^{bc} mod *p*, and Charlie and Alice share u^{ac} mod *p*. (Figure 4)

Step 2. Bob sends u^{ab} mod *p* to Charlie, who sends u^{bc} mod *p* to Alice, who sends u^{ac} mod *p* to Bob.

Step 3. Using their secret numbers, each is now able to compute the same key to be shared amongst them which is u^{abc} mod p. (Figure 5)

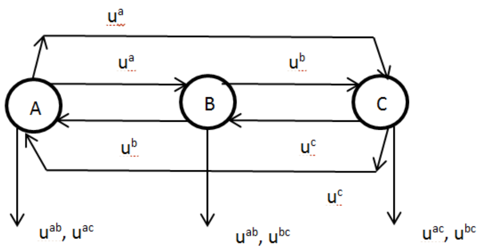

Figure 4. Pairwise exchange of random numbers

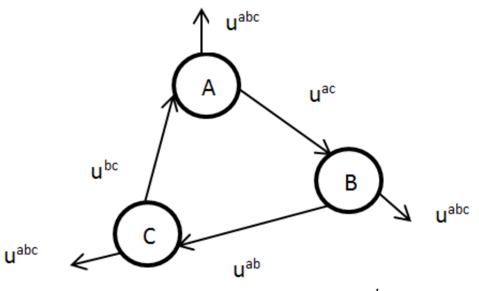

Figure 5. Generation of the single key u^{abc} mod p

As is clear from the working of this protocol as shown in Figures 4 and 5, the pairwise sharing of numbers as well as the final generation of a single number can be generalized to any number of parties.

If one wished to use this protocol to generate oblivious transfer then the parties should randomly choose between a set of potential bases as in Step 4.

Step 4. The three parties choose from different public numbers of larger order mod p. We will call these u, v, and w (if there are three such numbers).

Now consider that the base integers used by the three are two in number and let's call them u and v. If the secrets are exchanged in pairs then the probability that any two of them will share mutual secrets is ¼.

On the other hand, if there is a single secret that is coded by Alice using u^{abc} mod p, then there is a ¼ probability that both Bob and Charlie will receive it.

VI. VERIFICATION PROCESS FOR THREE BASE INTEGERS

Now consider that there are three base integers, u, v, and w. To forestall cheating by any party, one would need to develop a verification sequence by using a previously announced random number r that is used as an exponent on the respective raw keys.

Consider $G(n) = u^n + v^n + w^n$ mod p. To relate the three variables amongst each other, we need a quadratic expansion of the kind below:

$$\begin{aligned} u^3 &= \alpha u^2 + \beta u + \gamma \bmod p \\ v^3 &= \alpha\, v^2 + \beta\, v + \gamma \bmod p \\ w^3 &= \alpha\, w^2 + \beta w + \gamma \bmod p \end{aligned} \qquad (7)$$

This may be written down as the matrix equation:

$$\begin{bmatrix} u^3 \\ v^3 \\ w^3 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \\ v^2 & v & 1 \\ w^2 & w & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \lambda \end{bmatrix}$$

The solution of this equation is easily found to be:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{(v-w)(w-u)(u-v)} \begin{bmatrix} v-w & w^2-u^2 & v^2w-vw^2 \\ w-u & u^2-w^2 & uw^2-wu^2 \\ u-v & u^2-w^2 & u^2v-uv^2 \end{bmatrix} \begin{bmatrix} u^3 \\ v^3 \\ w^3 \end{bmatrix} \qquad (8)$$

Theorem 2.

$$G(n) = \alpha\, G(n-1) + \beta\, G(n-2) + \gamma G(n-3) \bmod p \qquad (9)$$

Proof. $G(n) = (u^n + v^n + w^n) \bmod p$

$$\begin{aligned} &= (u^{n-3}u^3 + v^{n-3}v^3 + w^{n-3}w^3) \bmod p \\ &= u^{n-3}(\alpha u^2 + \beta u + \gamma) + v^{n-3}(\alpha^2 v + \beta v + \gamma) \\ &\qquad + w^{n-3}(\alpha^2 w + \beta w + \gamma) \bmod p \\ &= \alpha G(n-1) + \beta G(n-2) + \gamma G(n-3) \bmod p \end{aligned}$$

The sum of successive powers of v and w suffices to establish that they have been computed to the same exponent. All that is required to find the values of α and β is the solution to equation (7) for $k = 2$. No knowledge of the actual value of n is needed while computing equation (9).

Example 3. Let u=2, v=3, and w=5 mod 17. To find α, β, and γ, we use equation (7), obtaining:

$$\alpha = 10; \beta = 3; \gamma = 13 \bmod 17$$

The series $G(n) = 2^n + 3^n + 5^n \bmod 17$, for n = 0, 1, 2, 3… is as follows: 3, 10, 4, 7, 8, 0, 13 …

for which each nth element is *10 G(n-1) +3G(n-2) +13G(n-3)* mod 17. For example, the value 13 is 10×0+3×8+13×7 mod 17.

VII. DISCUSSION

This paper reviewed the problem of generation of random events using classical and quantum techniques. It then presented a variation of the DH key exchange protocol to serve as an oblivious transfer protocol that can easily generate a probability event of 1/m, where m is 2 or higher integer. A verification procedure was presented that can catch attempts by Alice at cheating. This method was also extended to three or more parties and the specific protocol together with the verification algorithm was presented for three parties.

REFERENCES

[1] A Kolmogorov, Three approaches to the quantitative definition of information. Problems of Information Transmission. 1:1-17 (1965)

[2] S. Kak, Classification of random binary sequences using Walsh-Fourier analysis. IEEE Trans. on Electromagnetic Compatibility, EMC-13: 74-77 (1971)

[3] G. Chaitin, Randomness and mathematical proof. Scientific American. 232(5): 47-52 (1975)

[4] S. Kak and A. Chatterjee, On decimal sequences. IEEE Trans. on Information Theory IT-27: 647 – 652 (1981)

[5] G. Marsaglia, A current view of random number generators, in Computer Science and Statistics: The Interface. 3-10. Elsevier Science (1985)

[6] S. Kak, Encryption and error-correction coding using D sequences. IEEE Trans. on Computers C-34: 803-809 (1985)

[7] G. Marsaglia and L.H. Tsay, Matrices and the structure of random number sequences. Linear Algebra Appl. 67: 147-156 (1985)

[8] R. Merkle, Secure communications over insecure channels. Comm. Of the ACM 21(4): 294-299 (1978)

[9] R. Feynman, QED: The Strange Theory of Light and Matter. Princeton Univ Press (1985)

[10] R. Landauer, The physical nature of information. Phys. Lett. A 217: 188-193 (1996)

[11] S. Kak, The initialization problem in quantum computing. Foundations of Physics, 29: 267-279 (1999)

[12] S Kak, Quantum information and entropy. Int. Journal of Theo. Phys. 46: 860-876 (2007)

[13] I. Gerhardt, Q. Liu, A. Lamas-Linares, J. Skaar, C. Kurtsiefer, and V. Makarov, Full-field implementation of a perfect eavesdropper on a quantum cryptography system. Nat. Commun. 2: 349 (2011)

[14] C.H. Bennett, G. Brassard, Quantum cryptography: Public key distribution and coin tossing. Proceeding of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, India, pp. 175–179, IEEE, New York (1984)

[15] S. Kak, A three-stage quantum cryptography protocol. Foundations of Physics Letters 19: 293-296 (2006)

[16] M. Rabin, Digitalized signatures and public key functions as intractable as factoring. Tech. Rep. MIT/LCS/TR-212, MIT (1979)

[17] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts. Comm. of the ACM 28: 637-647 (1985)

[18] S. Kak, The cubic public-key transformation. Circuits Systems Signal Processing 26: 353-359 (2007)

[19] S. Singh, The Code Book: the Secret History of Codes and Code-breaking. FourthEstate, London (1999)

[20] S. Kak, The Loom of Time. DKPrintworld, New Delhi (2016)

[21] R. Stoneman, The Ancient Oracles. Yale University Press (2011)

[22] A.R. Burn, Herodotus: The Histories. Penguin Classics (1972)

[23] E. Evans-Pritchard, Witchcraft, Oracle, and Magic Among the Azande. Oxford University Press (1976)

Subhash Kak is Regents Professor in the School of Electrical and Computer Engineering at Oklahoma State University at Stillwater. He is the author of twenty books that include The Nature of Physical Reality (3rd edition Mississauga, Mt. Meru, 2016), The Architecture of Knowledge (New Delhi, Motilal Banarsidass, 2004), and Matter and Mind (Mississauga, Mt. Meru, 2016). His areas of interest include data security, quantum computing, information theory, neural networks, and history of science. Professor Kak's awards include British Council Fellow (1976), Science Academy Medal of the Indian National Science Academy (1977), Kothari Prize (1977), UNDP Tokten Award (1986), Goyal Prize (1998), National Fellow of the Indian Institute of Advanced Study (2001), and Distinguished Alumnus of IIT Delhi (2002).