Decrypting Network Traffic- Shared Access Control

K.P.Vidya, *Member, IAENG*

***Abstract*—In this paper we propose a design to develop an engineering device for shared access to the decryption key of a public key cryptosystem used for secure communication over a network. The mechanism of the device is based on cryptanalytic tools that are used to attack the Elliptic Curve Discrete Logarithm Problem (ECDLP). The device comprises of three components: Share Generator, Authenticator and Key Generator, and its significance lies in safeguarding the secret keys of a public key cryptosystem that is used to transmit messages across the communication network.**

***Index Terms*—ECDLP, Elliptic Curves, Pollards rho Attack on ECDLP, Secret Sharing, Threshold Schemes.**

I. Introduction

The digital world today has brought with it a great demand for information security products. The security aspects of these products are based extensively on the advancements in the science of Cryptology. The techniques used in these products to secure information emphasize on the intractability of mathematical problems such as the Elliptic Curve Discrete Logarithm Problem (ECDLP). The incorporation of such cryptographic schemes in devices ensures that data communicated across networks are not vulnerable to attacks by adversaries. That is, it ensures data authentication and integrity. But what is important is to safeguard the secret keys that are used to decipher the encrypted data.

Threshold schemes play a very important role in safeguarding cryptographic keys. In a (t, n) scheme, a secret piece of information can be shared among a group of n persons such that, any t of them, $t \leq n$ may pool in their shares to recover the secret while $(t-1)$ persons cannot. This security check on the number of persons involved in reconstructing the secret helps to distribute control of the access to the data communicated across a network. In our paper we give the design for a device that works on the principle of the Pollard rho attack on the *ECDLP*. The algorithm not only provides an effective method in generating shares to a secret that can be shared among n persons in a group, but also, provides a method to verify these shares and thus authenticate the m, $t \leq m \leq n$, participants involved in reconstructing the secret key. This (t, n)-threshold scheme that is embedded in the Key Access Device enhances the utility of the device with the replacement of only one component embedded in the device whenever the security needs demand a change in the secret key.

Section 2 and 3 of this paper gives a brief understanding of secret sharing schemes and the mathematical background that is required to develop the security system. Section 4 gives a brief introduction to our scheme and section 5 describes the Key Access Device (*KAD*), its design and the underlying mathematical algorithm with an illustration. Section 6 discusses the merits of implementing our cryptographic scheme in the *KAD*.

Manuscript received May 16, 2007.

K.P. Vidya is with the Department of Mathematics, Madras Christian College (Autonomous), Affiliated to the University of Madras, Chennai, 600 059, India. (phone: 0091-044-24919718; e-mail: kpvidya@ hotmail.com).

II. Secret Sharing Schemes

A secret sharing scheme is that in which a secret α is divided into n shares which are distributed among the n participants so that a coalition of authorized participants can combine to reconstruct the secret. Shamir's[25] results based on Lagrange's interpolation of polynomials simultaneously with Blakley's[3] contribution on geometric hyper-planes were the first ever known secret sharing schemes that were later classified as threshold schemes. A generalization of the scheme was proposed in [15], and [2] describes its relation to monotone functions.

A scheme is called a threshold secret sharing scheme with a threshold value of t if only a coalition of $t \leq n$ participants can reconstruct the secret while $t-1$ or fewer participants cannot. If Φ denotes the group of participants and Γ and Δ respectively denote the set of authorized and unauthorized participants where Γ and Δ are assumed to be mutually disjoint then the collection (Γ, Δ) is called the *access structure* of the secret sharing scheme. The access structure is called a *monotone* access structure if a set P containing Γ is also a set of authorized participants. A hierarchical threshold access structure [32] defines sets of participants distributed in say l levels with different or same threshold values for each level. The level zero indicates the central point, the lower levels are called level one, level two, etc.. If different access structures in a family of access structures are to be activated at different instances of time then we say that the secret sharing scheme is dynamic. A fully dynamic secret sharing scheme as defined in [6] is the sharing of a set of secrets among a group of participants such that any subset of participants has no information about the new secret before knowing the new broadcast message but there

exists a perfect secret sharing scheme after seeing the new broadcast message. The fully dynamic secret sharing scheme is said to be *strong* if any subset of participants not in the new access structure and who know all the previous secrets, still have no information about the new secret.

A *perfect* secret sharing scheme is one in which the shares corresponding to each unauthorized subset provides absolutely no information about the shared secret. In fact, they have a monotone access structure. The efficiency of any secret sharing scheme is measured by its information rate = (Size of the shared secret) / (size of that participant's share). Since in any perfect secret sharing scheme the size of a share is greater than or equal to the size of the shared secret for all shares of the participants of the scheme, it follows that all perfect secret sharing schemes have information rate ≤ 1. Secret sharing schemes of rate 1 are called *ideal*. The Shamir's scheme is an example of a perfect and ideal threshold scheme.

III. Mathematical Background

A. Elliptic Curves, ECDLP and Pollard's rho Attack on ECDLP

An elliptic curve E defined over a finite field F_q, of characteristic greater than three is given by the set of points that satisfy the equation $y^2 = x^3 + ax + b$, $a, b \in F_q$ where, discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$ together with the point at infinity O. It forms an abelian group over a special type of addition, where, O serves as the identity element of the group and the inverse of a point $R = (x_1, y_1)$ on the curve is given by $-R = (x_1, -y_1)$. The Group law for addition of two points $R = (x_1, y_1)$ and $S = (x_2, y_2)$ for $R \neq S$ and $S \neq -R$, is given by the co-ordinates $(x_3, y_3) \in E(F_q)$ where, $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$ and the slope λ is given by $(y_2 - y_1)/(x_2 - x_1)$ if $R \neq S$ and $S \neq -R$ and $(3x_1^2 + a)/2y_1$ if $R = S$. The order p of the elliptic curve over F_q, i.e., the number of elements in the abelian group is determined by the bounds stated in Hasse's Theorem $q + 1 - 2\sqrt{q} < p < q + 1 + 2\sqrt{q}$ while the order of a point $R \in E(F_q)$ is the smallest positive integer α for which $\alpha R = O$. Further, if the group is of prime order it implies that the group is cyclic and every element of the group other than O is a generator of the group.

Definition The elliptic curve discrete logarithm problem (ECDLP): Given an elliptic curve E defined over a finite field F_q, a point $P \in E(F_q)$ of order p, and a point $Q \in \langle P \rangle$, find the integer $l \in [0, p-1]$ such that $Q = lP$. The integer l is called the discrete logarithm of Q to the base P, denoted $l = \log_P Q$.

The Pollard's rho attack[22] on the *ECDLP* finds two distinct pairs (c', d'), (c'', d'') of integers modulo p such that the points $X' = c'P + d'Q$ and $X'' = c''P + d''Q$ collide. That is, a suitable iteration function $f: \langle P \rangle \rightarrow \langle P \rangle$ is defined so that any point X_0 in $\langle P \rangle$ determines a sequence $\{X_i\}_{i \geq 0}$ of points where $X_i = f(X_{i-1})$ for $i \geq 1$. Now, since $\langle P \rangle$ is finite, the sequence will collide at some i^{th} iteration and then cycle for the remaining iterations forming a ρ-like shape. Then l can be obtained by computing $l = (c' - c'')(d'' - d')^{-1} \bmod p$. This is in the case of a single processor which has a run time complexity of $\sqrt{(\pi n/2)}$ due to Teske[34] who suggested an iterating function in pollard's algorithm that modeled on a random walk.

IV. Our Scheme

Our scheme is a (t, n) threshold scheme which we call the Pollard Secret Sharing Scheme(Single Processor), where, α is set as the secret of the threshold scheme. $\langle P \rangle$ is partitioned into n number of sets of roughly the same size and these form the shares or shadows that are distributed to all the participants A_i, $i = 1,2, \ldots, n$. The threshold value t of the scheme is set depending on the minimum number of partitions (shares) required that would determine the computational feasibility of the secret within the specified time limit requirement of the application. Thus the threshold value t depends not only on the number of *partitions* but also on the *processing speed* of the machine used to compute the secret key.

The scheme has three main phases, the Share Generation, Authentication or Share Verification, and the Reconstruction of the Secret.

An entity T who plays the role of the trusted authority generates the public and private key pair. The plain text message that is encrypted using the public key is transmitted across the network to its destination node. The cipher text message can be decrypted using the private key only if a coalition of authorized participants at that node combines to reconstruct the secret decryption key. The security of our scheme relies on the hardness of solving the ECDLP on classical computers. We specify classical computers here, since there exists a polynomial time algorithm proposed by Shor[26] to solve the ECDLP on quantum computers. So we assume that the system network is setup such that it is infeasible to compute within a specified period of time, a solution to an ECDLP instance given less powerful resources. But given more powerful machines it is an easy problem to solve. Thus the strength of computation of the secret reflects on the security level of the cryptosystem. The application of this scheme is made use of to design the Key Access Device described below.

V. Key Access Device

The Key Access Device (*KAD*) is an engineering device that may be deployed at every terminal of a communication network that uses a common public key to encrypt data and shares the decryption key amongst a group of authorized agents at each terminal node. Its functionality is to derive the secret key of the public key cryptosystem from inputs received from authorized agents and transmits this secret key as input to an *Enc-Dec* device that encrypts and decrypts the network traffic. The shares that are jointly deposited into the *KAD* by any t or more number of participants from a group of n authorized persons at each node of a communication network are used to reconstruct the secret decryption key. This envisages a greater role for the

KAD in providing access to the secret key of a cryptosystem without compromise of the secret key to any of the authorised agents.

A trusted third party T generates a public and private key pair and publishes the public key so that anyone who wants to send a message can encrypt the plain text with the public key. The secret α to be used as the private key of the cryptosystem is shared among n participants at each centre C_i in the communication network. T generates the verification parameters P and Q that are points on an elliptic curve E of order p such that $Q = \alpha P$. These parameters P, Q and p are stored in a chip at the time of manufacture together with a unique device identity number *DID*. The *DID* is specific to each Key Access Device (*KAD*) that is installed at the different centres C_i of the communication network. The chips are parceled to the respective centres where the authorised agents at these locations, insert the chip into the *KAD* to generate their shares of the secret decryption key. This generation of the shares of the secret key can be carried out only once in the lifetime of the chip. Each share also consists of a unique value that authenticates each authorized agent at the time of reconstruction of the secret key. These unique values are stored in the memory chip of the Share Generator component of the *KAD* to be used at the time of reconstruction of the secret key.

The *KAD* now operates to receive the shares of any m, $t \le m \le n$ authorised agents whenever the encryption-decryption device (*Enc-Dec*) that is connected to the computer terminal (*CT*) prompts for data decryption. The Authenticator component of the *KAD* verifies the authenticity of these shares that are input by the authorised agents. It then creates a data block *DB* that is transmitted to the third component of the *KAD* called the Key Generator. Here, the secret α is constructed using the information in *DB* which is then delivered to the encryption-decryption device. On receiving the secret α from the *KAD* the *Enc-Dec* decrypts the message based on the request received from the terminal.

A. *Components and their functions*

The Key Access Device comprises of two parts a main device called the *KAD* and a detachable chip *D*. The Share Generator, Authenticator, and Key Generator form the three components of the *KAD*.

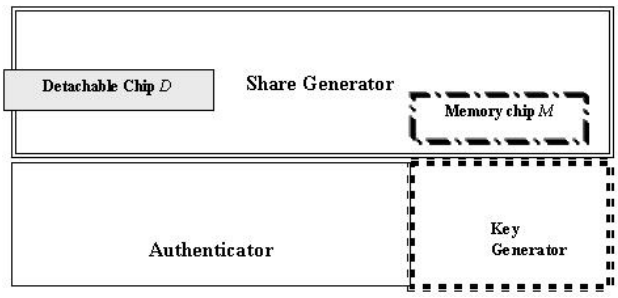

Fig.1. Components of the Key Access Device (*KAD*)

Share Generator

The Share Generator has two sub components, a detachable chip *D* and a fixed programmable memory chip *M*. The unique device identity number *DID*, the verification parameters *P* and *Q*, and the order *p* of the elliptic curve group are stored in *D* at the time of manufacture. When *D* is inserted into the Key Access Device, the function *g* displays *n* and *p*, and receives *n* sets of inputs of the Agents' choice of random integers, a_i, b_i which belong to the interval $[1, p-1]$. It then computes the respective unique value R_i a point on the Elliptic Curve *E* for each Agent and stores these R_i in the memory chip *M* as the identity value of the respective agent. These are used later for authenticating the agents at the time of generating the secret key. However, this process is carried out only if the detachable chip *D* and the *KAD* are found to be compatible. This compatibility of the *D* with the main device is verified by comparing the *DID* of the detachable chip *D* with that of *KAD*.

Authenticator

When the *Enc-Dec* device prompts for the secret key the Authenticator beeps loudly for the shares of the agents where, any *m* agents, $t \le m \le n$, need to combine to reconstruct the key. Let us suppose that the Authenticator receives the inputs say, (a_j, b_j) from Agents A_j , $j = 1, \ldots, m$, respectively. The function g' accesses the verification parameters *P* and *Q* from the chip *D* of the share generator and computes $V_j = a_jP + b_jQ$, $j = 1, \ldots, m$, respectively. If these values of V_j are found to be equal to some R_j in the memory chip *M* of the share generator, the authenticator proceeds to construct a data block *DB* that is passed as input to the next component the Key Generator. Otherwise, the agents A_j are denied access to decrypt the messages transmitted across the network.

The data block *DB* comprises of the shares (a_j, b_j, R_j) of the agents A_j , $j = 1, \ldots, m$, respectively, the verification parameters *P* and *Q* and the order *p* of the elliptic curve group $E(F_q)$. This data block is passed as input to the Key Generator component to compute the secret key α.

Key Generator

The process block g'' in this component receives the data block *DB* from the Authenticator and outputs the secret $S = \alpha$. At first, the initial values of the iterative process are computed as $c' = \Sigma\, a_j$ and $d' = \Sigma\, b_j$ and $X' = \Sigma\, R_j = c'P + d'Q$. The operations are carried out modulo *p*. Then, a set *L* is formed in which each element of the set acts as an index to the partitions or shares in *DB*. A partition function *H* defined from $<P>$ to *L* determines for $X' \varepsilon <P>$, the value $h = H(X') = H(x, y) = x \bmod m + 1$. Here, *m* is the cardinality of *L*. Now, the variable *h* assumes a value that is an element of set *L*. In the iterative process, the shares are chosen corresponding to the value of *h* and the process of computing X' and $X''$is repeated till their values are found to coincide. The secret $S = \alpha$ is then obtained as $\alpha = (c' - c'')(d'' - d')^{-1} \bmod p$.

B. Design

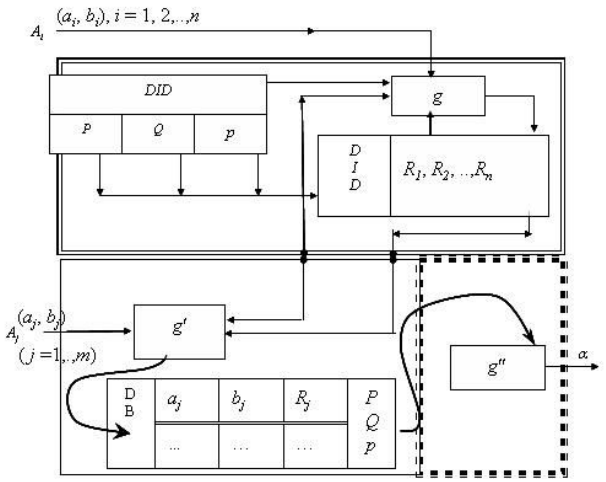

Fig.2. Design of the Key Access Device

C. Mechanism

The Pollard Secret Sharing Scheme (Single Processor)

SUMMARY A secret key α used in a cryptosystem is distributed among n participants A_i. $i = 1, \ldots, n$, of the (t, n)-threshold scheme.

RESULT Any m participants for $t \le m \le n$ pool in their shares to reconstruct the secret.

I Share Generator:

1. E is the chosen elliptic curve over a finite field F_q generated by $<P>$ of prime order p.
2. The secret α that controls the critical action is a random integer l and determines the point $Q = lP$ on E.
3. Random integers $a_i, b_i \in [1, p-1]$ are chosen such that $R_i = a_iP + b_iQ$, $i = 1 \ldots n$.
4. $S_i = (a_i, b_i)$ are the shares of the participants A_i, $i = 1, \ldots, n$
5. (P, Q) the verification parameters and the prime p are embedded in the chip D.

II Authenticator (in case of m inputs where $t \le m \le n$)

1. Any m number of participants, say, A_j, $j = 1, 2, \ldots m$, pool-in their shares, $t \le m \le n$.
2. Verification parameters P and Q are accessed from the Share Generator to compute
$$a_jP + b_jQ = V_j, \quad j = 1, 2, \ldots m$$
and compare if V_j equals respective R_j.
3. If $V_j = R_j$, the step 4 and the steps in Reconstruction of Secret is carried out.
4. Set the Data Block DB with the shares (a_j, b_j, R_j), $j = 1, 2, \ldots m$ and P, Q, p.

III Key Generator

1. Receive Data Block DB from *Authenticator*.
2. Set $L = \{1, 2, \ldots, m\}$.
3. Set $H: <P> \rightarrow L = \{1, 2, \ldots, m\}$ a partition function where m indicates the number of partitions that are used during recovery of secret. Here, we choose a simple partition function such that, for $X' \in <P>$, $h = H(X') = H(x, y) = x \bmod m + 1$ for $t \le m \le n$.
4. Set $c' = \Sigma a_j \pmod p$, $d' = \Sigma b_j \pmod p$ and $X' = \Sigma R_j = c'P + d'Q \pmod p$.
5. Repeat
   a) Compute $h = H(X')$ where h corresponds to an element in L.
   b) Set $X' = X' + R_h \pmod p$, $c' = c' + a_h \bmod p$, $d' = d' + b_h \bmod p$.
   c) For r from 1 to 2 do
      i) Compute $h = H(X'')$, where h corresponds to an element in L.
      ii) Set $X'' = X'' + R_h \pmod p$, $c'' = c'' + a_h \pmod p$, $d'' = d'' + b_h \pmod p$.

   Until $X'' = X'$.
6. Compute $l = (c' - c'')(d'' - d')^{-1} \bmod p$ which is the secret α. When reconstruction of α takes place the *KAD* sends this α to the Enc-Dec device for decryption of the message received across the network.
7. Exit.

D. Illustration

Suppose that, A_i, $i = 1, \ldots, 5$, are the participants of a secret sharing scheme and that a subset of two or more participants are to combine to reconstruct the secret key of the cryptosystem.

Share Generator: The trusted entity T selects at random the elliptic curve $E(F_{29})$ given by $y^2 = x^3 + 4x + 20$ where the discriminant $\Delta = -176896 \not\equiv 0 \pmod{29}$. The number of elements in the elliptic curve group is 37 a prime, and so, $E(F_{29})$ is a cyclic group. All elements of $E(F_{29})$ for $P = (1, 5)$ as the generator are listed in the **Table 1** below. Now assume that, the secret S is set as equal to 30. If the point P in our algorithm is chosen to be the pair (1, 5) then $Q = 30P = (24, 7)$.

For inputs a_i, b_i from the authorized agents the shares are computed as $S_i = (a_i, b_i, R_i)$ for i equal to 1 to 5. If the corresponding shares for each A_i are $S_1 = (28, 34, (19, 13))$, $S_2 = (17, 27, (16, 27))$, $S_3 = (20, 14, (15, 2))$, $S_4 = (14, 23, (1, 5))$ $S_5 = (12, 3, (14, 6))$, the participants A_i, $i = 1, \ldots, 5$, retain the knowledge of the pairs of random integers (a_i, b_i) and the unique R_i are stored in the chip M to authenticate each A_i at the time of reconstructing the secret key.

Authenticator: Now, suppose that, A_1, A_3 and A_5 wish to determine the secret key to decrypt the cipher text received at their computer terminal C_i in the network. When the agents input their random number pairs the respective value of R_i are computed and the shares are set as $S_1 = (28, 34, (19, 13))$, $S_3 =$

(20 , 14 , (15, 2)) and S_5 = (12 , 3 , (14, 6)) after verification of their authenticity.

Table 1.

0P=O	13P=(16,27)	26P=(10,4)
1P= (1,5)	14P=(5,22)	27P=(13,6)
2P= (4,19)	15P=(3,1)	28P=(14,6)
3P= (20,3)	16P=(0,22)	29P=(8,19)
4P=(15,27)	17P=(27,2)	30P=(24,7)
5P=(6,12)	18P=(2,23)	31P=(17,10)
6P=(17,19)	19P=(2,6)	32P=(6,17)
7P=(24,22)	20P=(27,27)	33P=(15,2)
8P=(8,10)	21P=(0,7)	34P=(20,26)
9P=(14,23)	22P=(3,28)	35P=(4,10)
10P=(13,23)	23P=(5,7)	36P=(1, 24)
11P=(10,25)	24P=(16,2)	
12P=(19,13)	25P=(19,16)	

Key Generator: The initial values of the iterating function are given by (c', d', X') = (23, 14, (1, 24)) where c', d' ε[0, 36] and $X' = c'P + d'Q = 23P + 14(30P) = 36P$ modulo 37 = (1, 24). The set L is set as L = {1, 2, 3}. The tabulations of c', d', X', c'', d'', X'' for the iterations are shown in **Table 2**. The process terminates in the 4th iteration when X' = X'' = 31P. The corresponding values of c', d', c'', d'', are 16, 19, 7, 23 respectively.

Table 2

Itr	c'	d'	X'	c''	d''	X''
--	23	14	36P = (1, 24)	23	14	36P = (1, 24)
1	6	28	32P = (6, 17)	25	22	19P = (2, 6)
2	34	25	7P = (24, 22)	19	10	23P = (5, 7)
3	25	22	19P = (2, 6)	13	35	27P = (13, 6)
4	16	19	31P = (17, 10)	7	23	31P = (17, 10)

Now, $l = (16 - 7)(23 - 19)^{-1} \pmod{37} = 30$ gives the value of the secret α. On the reconstruction of the secret α the critical action is carried out by A_1, A_3 and A_5.

VI. Merits

The Key Access Device plays a significant role in safeguarding the secret keys of a cryptosystem irrespective of the device and the technique used for the encryption and decryption process. Our security technique offers a simple method to update the secret keys without compromising the keys themselves. The use of a detachable chip to update the keys increases the functionality of the device with a need to replace only one of its components. Also, since the agents at each centre are given a share that can be verified, a log file generated by the device can ensure non-repudiation in case of any controversy involved in the communication process.

Our scheme based on the Pollard rho attack on ECDLP also offers a very efficient iterative function that requires negligible memory space. A better choice of the pair (H, f) where, H is the hash function that determines a point in a partition and f is the iteration function that determines the sequence of points in the elliptic curve that collide at some stage, results in better random walks that give added advantages in the speed up of the algorithm.

Since the compromise of the secret key used in a cryptosystem poses a serious security threat it becomes essential that the system is periodically replaced with new systems. But the implementation of our scheme in the Key Access Device minimises this threat and hence reduces the expenses incurred in frequent replacement of cryptosystems.

References

[1] Benaloh, J. C., *Secret sharing homomorphisms: keeping shares of a secret secret*, in Advances in Cryptology -- CRYPTO '86, A. M. Odlyzko, ed., Lecture Notes in Computer Science 263 (1987), 251-260.

[2] Benaloh, J. C., Leichter, J., *Generalized Secret Sharing and Monotone Functions*. Proceedings of CRYPTO 1988: 27-3.

[3] Blakley, G.R., *Safeguarding cryptographic keys*, In Proc. Nat. Computer Conf. AFIPS Conf. Proc., pp. 313-317, 1979.vol48.

[4] Blakley, B., Blakley, G.R., Chan, A.H., and Massey, J., *Threshold Schemes with Disenrollment*, Advances in Cryptology – CRYPTO '92, E.Brickell, Editors, Lecture Notes in Computer Science, Springer-Verlag.

[5] Blundo, C., *A note on dynamic threshold schemes*, Information Processing Letters, 55 (1995) 189-193.

[6] Blundo, C., Cresti, A., De Santis, A., Vaccaro, U., *Fully Dynamic Secret Sharing Schemes*, Theoretical Computer Science **155** (1996), 407-410.

[7] Blundo, C., De Santis, A., Stinson, D. R., and Vaccaro, U., *Graph decompositions and secret sharing schemes*, Journal of Cryptology 8 (1995), 39-64.

[8] Brickell, E.F., Davenport, D.M., *On the classification of ideal secret sharing schemes*, Journal of Cryptology 4 (1991) 123-134.

[9] Capocelli, R. M., De Santis, A., Gargano, L., and Vaccaro, U., *On the size of shares in secret sharing schemes*, Journal of Cryptology 6 (1993), 157-167.

[10]Cramer, R., Gennaro, R., and Schoenmakers, B., *A Secure and Optimally Efficient Multi-Authority Election Scheme*, European Transactions on Telecommunications, vol. 8, no. 5, pp. 481-490, Sep 1997.

[11] Desmedt, Y., *Society and group oriented cryptography: a new concept*. In C. Pomerance, editor, Advances in Cryptology, Proc. Of Crypto '87 (Lecture Notes in Computer Science 293), pp.120-127. Springer-Verlag. 1988. Santa Barbara, California, U.S.A., August 16-20.

[12] Desmedt, Y., *Threshold cryptography*, In W. Wolfowicz, editor, Proceedings of the 3rd Symposium on: State and Progress of Research in Cryptography, pp. 110-122, February 15-16, 1993. Rome, Italy, invited paper.

[13] Fouque, P.A., Poupard, G., and Stern, J., *Sharing Decryption in the Context of Voting or Lotteries*, Financial Cryptography 2000.

[14] Gallant, R., Lambert, R., and Vanstone, S., *Improving the parallelized Pollard Lambda search on anomalous binary curves*, Mathematics of Computation, 69:1699-1705, 2000.

[15] Ito, M., Saito, A., Nishizeki, T., *Secret sharing scheme realizing any access structure*, Proc. IEEE Globecom'87 (1987) 99-102.
[16] Karnin, E. D., Greene, J. W., and Hellman, M. E., *On secret sharing systems*, IEEE Transactions on Information Theory 29 (1983), 35-41.
[17] Koblitz, N., Elliptic curve cryptosystems, Math. Comp., 48(177) : 203-209, January 1987.
[18] Koblitz, N., *Algebraic aspects of cryptography*, Algorithms and Computations in Mathematics, Volume 3, Springer 1999.
[19] Kuhn, K., Struik, R., *Random walks revisited: Extensions of Pollard's rho algorithm for computing multiple discrete logarithms*, Selected Areas in Cryptography-SAC 2001 Lecture Notes in Computer Science 2259, Editors-S.Vaudenay and A.Yousef, 212-229, 2001.
[20]Kurosawa, K., Obana, S., and Ogata, W., *t-cheater identifiable (k, n) threshold secret sharing schemes*, in "Advances in Cryptology -- CRYPTO '95", D. Coppersmith, ed., Lecture Notes in Computer Science 963 (1995), 410-423.
[21] Oorschot van, P., Weiner, M., *Parallel collision search with cryptanalytic applications*, Journal of Cryptology, 12: 1-28, 1999.
[22] Pollard, J.M., *Monte Carlo methods for index computation mod p*, Math. Comp., 32(143): 918-924, July 1978.
[23] Sedgwick, R., Symanski, T., and Yao, A., *The complexity of finding cycles in periodic functions*, SIAM Journal on Computing 11: 376-390, 1982.
[24] Seymour, P.D., *On secret-sharing matroids*, Journal of Combinatorial Theory Ser. B, 56 (1992) pp. 69–73.
[25] Shamir, A., *How to share a secret*, Communications of ACM, 22, pp. 612-613, November 1979.
[26] Shor, P.W., *Algorithms for quantum computation: discrete log and factoring*, in Proceedings of the 35th Annual Symposium on the Foundations of Computer Science (IEEE Computer Society, Los Alamitos, 1994), p. 124.
[27] Silverman, J.H., The Arithmetic of elliptic curves, volume 106 of Graduate Texts in Mathematics, Springer-Verlag, 1986.
[28] Simmons, G.J., *How to (really) share a secret*, Santa Barbara, California, U.S.A., Advances in cryptology, Proc. of Crypto '88 (Lecture Notes in Computer Science) Springer-Verlag, August 1988.
[29] Simmons, G. J., *Prepositioned shared secret and/or shared control schemes*, in "Advances in Cryptology -- EUROCRYPT '89", J.-J. Quisquater and J. Vandewalle, eds., Lecture Notes in Computer Science 434 (1990), 436-467.
[30] Smart, N., *The discrete logarithm problem on elliptic curves of trace one*, Journal of Cryptology, 12:193-196, 1999.
[31] Stinson, D. R., *Decomposition constructions for secret sharing schemes*, IEEE Transactions on Information Theory 40 (1994), 118-125.
[32] Tassa, T., *Hierarchical threshold secret sharing*, M.Naor, editor, Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Volume 2951 of lecture notes in Computer Science, p.473-490, Springer-Verlag, 2004.
[33] Teske, E., *Speeding up Pollard's rho method for computing discrete logarithms*, Algorithmic Number Theory-ANTS-III (Lecture Notes in Computer Science 1423) [82], 541-554, 1998.
[34] Teske, E., *On random walks for Pollard's rho method*. Mathematics of Computation, 70: 809-825, 2001.
[35] Tompa, M., and Woll, H., *How to share a secret with cheaters*, Journal of Cryptology 1 (1988), 133-138.
[36] Weiner, M., and Zuccherato, R., Faster attacks on elliptic curve cryptosystems, Selected Areas in Cryptography-SAC 1998, Lecture Notes in Computer Science 1556, 190-200, 1999.