2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

Online Secret Sharing Using Infinite Convergent Sequences

Abdolreza Rasouli [1] and Mahboubeh Shamsi [2+]

[1, 2] Islamic Azad University, Bardsir, Kerman, Iran

Abstract. Multi Party Computation (MPC) is a mathematical cryptographic technique that securely computes a function over distributed data among two (2PC) or number of parties. It is secure and preserves the privacy of parties' data and do not reveal the sensitive data except the function's result. Multi party computations use a shared secret which is apportioned between all parties to ensure the privacy of the data. The shared secret guarantees that the function only is computable over all parties. The method of sharing the secret makes the algorithms to behave offline and inappropriate for online problems. In this paper, the online secret sharing problem is solved by convergent infinite product sequences. The aim of this research is to design a new innovative online secure multi party computation algorithm in case of online parties.

Keywords: Multi Party Computation, Secret Sharing, Infinite Sequences, Cryptographic Computation.

1. Introduction

Every day large volume of information produces and stores among multi parties systems. Although these data are produced by companies unrelated to each other and are stored in various parties, but when they are gathered together much valuable information and patterns reveals. Data mining over distributed data discovers this costly knowledge. However, ownership of data by different companies and maintain confidentiality of data is the main challenge. Secure Multi Party Computation is a set of methods that perform mathematic computation over the multi party distributed data with ensuring the privacy preserving of the confidential data. Most of these methods use a shared secret key to ensure the privacy of each party. All parties should be present to share the secret keys, but unfortunately in many real cases, the parties are not joining the process at the start time. The main problems addressed in this research are the online secret sharing. The infinite convergent product sequences are employed to overcome the dependency problem between the shared secret key and users' public keys, which make the algorithm runs offline. The designed online secret sharer allows the parties to join the system during process life.

Due to the nature of secret sharing, most of the existing multi party computation algorithms are offline and the rest are semi-online. There are two main problems to change the offline multi party computation algorithms to an online mode. The first problem is how a secret could share online among all parties. The shared secret is usually a calculation of the all parties' keys and therefore, all parties should present before the process starts to share their secret key. It means that no new party can join the system after the process started. The offline secret sharing problem avoids the algorithm to be applicable in online cases such as online e-voting, e-bidding or web polls.

2. Background and Related Work

There are several proposed solutions to secure multi party computation such as The Trusted Third Party [1,2,3], The Oblivious Protocol [4], 1-Out Of N Oblivious Protocol [5,6,7], Oblivious Evaluation Of Polynomials [8], Threshold Cryptography [7,9] and Secret Sharing [10,11,12,13]. Some other techniques like

+ [1] rs.reza@gmail.com, [2] mahboubehshamsi@yahoo.com

Secure Sum, Secure Set Union, Secure Size of Set intersection, Scalar Product, EM clustering can also be employed along with above mentioned approaches to find the SMC solutions [7].

Secret sharing is the method of sharing a secret by multiple parties, so that no one and no party know the secret, but the secret could be constructed by combing some parties' shares. A multi-secret sharing scheme is a protocol to share a number of (arbitrarily related) secrets among a set of participants in such a way that only qualified sets of participants can recover the secrets [11].

For example, in a two-party case, Alice and Bob share a value x modulo some appropriate value N, in such a way that Alice holds a, Bob holds b, and x is equal to $(a+b)$ *mod* N. This is called additive secret sharing. An important property of this kind of secret sharing is that if Alice and Bob have shares of a and b, then they can each locally add their shares modulo N to obtain shares of $a+b$.

Shamir secret sharing is a threshold scheme [10]. In Shamir secret sharing, there are N parties and a polynomial P of degree k-1 such that $P(0)=x$ where x is a secret. Each of the N parties holds a point in the polynomial P. Because k points (x_i, y_i) $(1 \leq i \leq k)$ uniquely define a polynomial P of degree k-1, a subset of at least k parties can reconstruct the secret x. But, fewer than k parties cannot construct the secret x. This scheme is also called (N, k) *Shamir secret sharing*.

The shared secret usually computes from a combination of all participants' public keys, so all parties should present their portion of the shared secret to create a shared key. The process of gathering all participants' portion of shared secret key tends to the offline secret sharing. The offline secret sharing process is displayed in Fig.1.a.

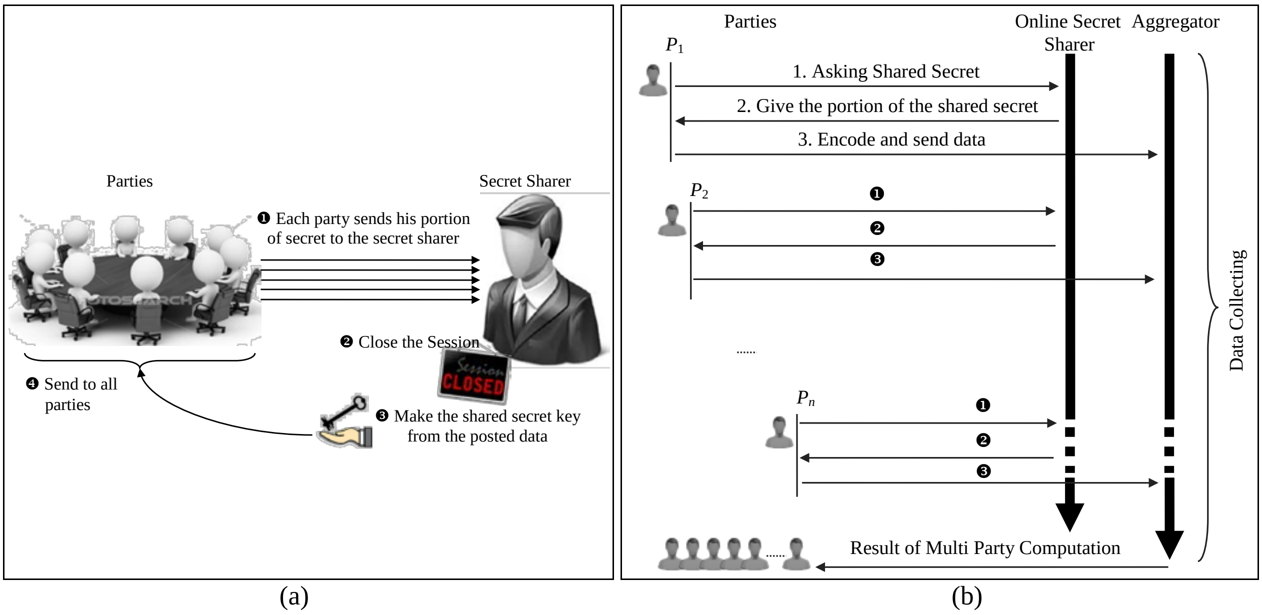


Fig. 1. a) Offline Secret Sharing - b) Online secret sharing scheme

When the secret shares offline, then the multi party computation executes offline due to the fundamental role of the secret sharing in multi party computation process. The offline secret sharing includes four stages;

1. Each party send his portion of the final shared secret key, usually his public key
2. When all parties send their portion of data, the secret sharer closes the session.
3. Secret sharer makes the shares secret key with combination of the posted data.
4. Secret sharer sends the shared secret key to all participants to enable them for joining the multi party computation process.

There are two types of secret sharers; in the first manner, the secret sharer is a trusted third party and in the second way, each party computes the shared secret himself. Despite the type of the secret sharer, the dependence between the shared secret and the participants, leads to an offline secret sharing process [14,15].

3. Online Secret Sharing

In these cases, the clients own their private and confidential data and the computing engine want to know the summation of all data without revealing the clients' secret data. But the clients are not ready at the beginning time and the time of user's appearance is optional. The computing engine also does not have any knowledge that how many clients could join to the system and when they will appear. So the secret sharer cannot compute the secret shared between the clients. The secret shared between the users is the product of their public keys as called *X*, *Y*. All clients should be present their public keys to compute the shared secret key *X*, *Y* before aggregation started. Moreover, all users should send their data; therefore, the computing engine could compute the summation. The online secret sharing scheme is shown in Fig.1.b.

In this model, the secret sharer is online during the process life and the users can join the system any time and ask the shared secret key. This model requires the independency between the shared secret key and users' public keys. While the participants receive the shared secret key, they try to encrypt their sensitive data with their public and private keys in addition to the shared secret key. The encrypted data is posted to the computing engine to be aggregated with others. At the end of the process, the result posts to all parties by the aggregator.

3.1. Online Secret Sharing Algorithm.

An infinite convergent product series is used as shared secret key instead of *X*, *Y* to establish the independency between the shared secret key and the participants' public keys. The final amounts of these series do not change with increasing the number of elements, so they ensure the unlimited number of clients who can join the process. An infinite product series define as

$$A = \prod_{i=1}^{\infty} \alpha_i \qquad (1)$$

like as

$$\prod_{i=1}^{\infty} \frac{1}{e} \left(\frac{1}{3i+1}\right)^{3i+\frac{1}{2}} = 1.0123785 \qquad (2)$$

The algorithm also is semi coded below.

```
Input: Each party $P_i$ has two pair of the public and private keys ($x_i$, $X_i = g^{x_i}$) and ($y_i$, $Y_i = g^{y_i}$) and his vital data $d_i$.
Outpout: Summation of all vital data $d=\sum_{i=1}^{n} d_i$.
Step 1. Made by Online Secret Sharer
Choose two convergent infinite product series as the shared secret
        $(\Omega_i mod_q)=(g^{\omega_i} mod_q)$, $(\Omega\ mod_q)=\prod_{i=1}^{n}(\Omega_i mod_q)$
        $(\Psi_i mod_q)=(g^{\psi_i} mod_q)$,     $(\Psi\ mod_q)=\prod_{i=1}^{n}(\Psi_i mod_q)$
Online secret sharer sends $\Omega$, $\Psi$ to joint party
Step 2. Sending data by the Participants
Each party encrypt his vital data
        $E(d_i)=\left\{\begin{matrix}(m_i\ mod_q)=(g^{d_i}\ mod_q).(\Omega^{y_i}\ mod_q)\\ (h_i\ mod_q)=(\Psi^{x_i}\ mod_q)\end{matrix}\right\}$
Send $m_i$ , $h_i$ to the aggregator.
Step 3. Computing the aggregation result by aggregator
Aggregator computes X, Y, m, h
  $(X\ mod_q)=\prod_{i=1}^{n}(X_i mod_q)$,$(Y\ mod_q)=\prod_{i=1}^{n}(Y_i mod_q)$, $m=\prod_{i=1}^{n}(m_i\ mod_q)$, $h=\prod_{i=1}^{n}(h_i\ mod_q)$
For d = 1 to Max(d) do
        If $(m\ mod_q).(X^{\sum_{i=1}^{n}\psi_i})=(g^d\ mod_q).(h\ mod_q).(Y^{\sum_{i=1}^{n}\omega_i})$ then
                return (d).
```

4. Online Secret Sharing Algorithm Demonstration

Suppose that $\left(m\, mod_q\right).(X^{\sum_{i=1}^{n}\psi_i}) = \left(g^d\ mod_q\right).\left(h\, mod_q\right).(Y^{\sum_{i=1}^{n}\omega_i})$

$$\Rightarrow \left(\prod_{i=1}^{n}(m_i\ mod_q)\right).\left(X^{\sum_{i=1}^{n}\psi_i}\right) = \left(g^d\ mod_q\right).\left(\prod_{i=1}^{n}(h_i\ mod_q)\right).(Y^{\sum_{i=1}^{n}\omega_i})$$

$$\Rightarrow \left(g^d\ mod_q\right) = \frac{(\prod_{i=1}^{n}(m_i\, mod_q)).\left(X^{\sum_{i=1}^{n}\psi_i}\right)}{(\prod_{i=1}^{n}(h_i\, mod_q)).\left(Y^{\sum_{i=1}^{n}\omega_i}\right)} = \frac{(\prod_{i=1}^{n}(g^{d_i}\, mod_q).(\Omega^{y_i}\, mod_q)).(X^{\sum_{i=1}^{n}\psi_i})}{(\prod_{i=1}^{n}(\Psi^{x_i}\, mod_q)).(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= \frac{(\prod_{i=1}^{n}(g^{d_i}\, mod_q).\prod_{i=1}^{n}(\Omega^{y_i}\, mod_q)).(X^{\sum_{i=1}^{n}\psi_i})}{(\prod_{i=1}^{n}(\Psi^{x_i}\, mod_q)).(Y^{\sum_{i=1}^{n}\omega_i})} = \prod_{i=1}^{n}\left(g^{d_i}\ mod_q\right).\frac{\prod_{i=1}^{n}(\Omega^{y_i}\, mod_q)}{\prod_{i=1}^{n}(\Psi^{x_i}\, mod_q)}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i\, mod_q)}.\frac{\Omega^{\sum_{j=1}^{n}\left(y_j\, mod_q\right)}}{\Psi^{\sum_{j=1}^{n}\left(x_j\, mod_q\right)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})} = g^{\sum_{i=1}^{n}(d_i\, mod_q)}.\frac{(\prod_{i=1}^{n}(\Omega_i mod_q))^{\sum_{j=1}^{n}\left(y_j\, mod_q\right)}}{(\prod_{i=1}^{n}(\Psi_i mod_q))^{\sum_{j=1}^{n}\left(x_j\, mod_q\right)}}.\frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i\,mod_q)} . \frac{\left(\prod_{i=1}^{n}(g^{\omega_i} mod_q)\right)^{\sum_{j=1}^{n}(y_j\,mod_q)}}{\left(\prod_{i=1}^{n}(g^{\psi_i} mod_q)\right)^{\sum_{j=1}^{n}(x_j\,mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})} = g^{\sum_{i=1}^{n}(d_i\,mod_q)} . \frac{g^{\sum_{i=1}^{n}(\omega_i\,mod_q).\sum_{j=1}^{n}(y_j\,mod_q)}}{g^{\sum_{i=1}^{n}(\psi_i\,mod_q).\sum_{j=1}^{n}(x_j\,mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i\,mod_q)} . \frac{g^{\sum_{j=1}^{n}(y_j\,mod_q)^{\sum_{i=1}^{n}(\omega_i\,mod_q)}}}{g^{\sum_{j=1}^{n}(x_j\,mod_q)^{\sum_{i=1}^{n}(\psi_i\,mod_q)}}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})} = g^{\sum_{i=1}^{n}(d_i\,mod_q)} . \frac{\left(\prod_{i=1}^{n}(g^{y_i}\,mod_q)\right)^{\sum_{i=1}^{n}(\omega_i\,mod_q)}}{\left(\prod_{i=1}^{n}(g^{x_i}\,mod_q)\right)^{\sum_{i=1}^{n}(\psi_i\,mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$= g^{\sum_{i=1}^{n}(d_i\,mod_q)} . \frac{\left(\prod_{i=1}^{n}(Y_i\,mod_q)\right)^{\sum_{i=1}^{n}(\omega_i\,mod_q)}}{\left(\prod_{i=1}^{n}(X_i\,mod_q)\right)^{\sum_{i=1}^{n}(\psi_i\,mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})} = g^{\sum_{i=1}^{n}(d_i\,mod_q)} . \frac{Y^{\sum_{i=1}^{n}(\omega_i\,mod_q)}}{X^{\sum_{i=1}^{n}(\psi_i\,mod_q)}} . \frac{(X^{\sum_{i=1}^{n}\psi_i})}{(Y^{\sum_{i=1}^{n}\omega_i})}$$

$$\Rightarrow \left(g^{d}\,mod_q\right) = g^{\sum_{i=1}^{n}(d_i\,mod_q)} \quad \therefore d = \textstyle\sum_{i=1}^{n} d_i \qquad \blacksquare \qquad (3)$$

The above demonstration proves that the represented algorithm correctly computes the summation of respondents' data and found d is the $\sum_{i=1}^{n} d_i$.

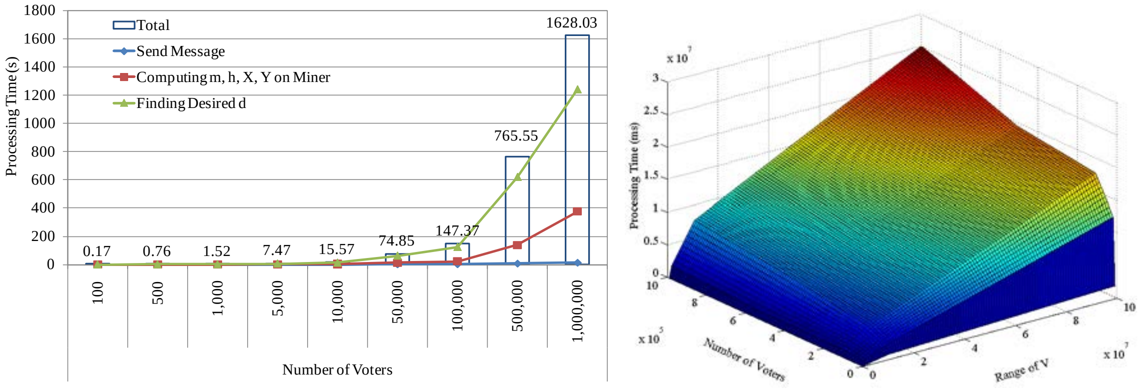
5. E-Voting Experimental Result Using the Online Secret Sharing

Suppose that, there is n voter and they have their own private votes. The voters can join to the election process any time they want. The Teller system should calculate the total of respondents' votes. Obviously the teller is not able to compute the exact vote of the respondents. The main parameters of experiment are: Number of Voters and the range of total votes V.

The algorithm's main stages are same as previous experiment, except that the X, Y should compute in Teller system instead of Precomputing offline. The stages are:

- Sending Encrypted message to Teller
- Computing m, h, X, Y on Teller
- Finding desired d across its' possible values

In first phase, each voter owns a Boolean data. In this case there are two options for voters. They can vote their desired item or reject it. In sooth, the algorithm counts the users' votes not sum them. The Teller aims to know how many users do vote the idea. In this experiment, the respondents send the Boolean data to Teller. In this condition, the range of *V* is equal to the number of voters. A range between 100 and 1,000,000 users are contributed in this experiment and the earned time is base on the average of five algorithm runs. The time is highly related to the number of users. For example, Teller computation takes 27 minutes for 1 million voters. The result is also shown in Fig.2.a.



(a) (b)

Fig. 2. a) Total time of Algorithm execution related to the Number of respondents – b) Total time related to Number of Voters and Range of V – 3d view

Second phase is dedicated to non Boolean cases. Suppose that a web administrator wants to know his visitors' opinion about a subject. He asks them about site attractive and wants them to vote from weak, not bad, good, attractive and high attractive. Each option is weighted from 1 to 5. The aim of administrator is to calculate the average points of visitors' opinion. In fact the web poll should sum all votes and then divide it to the number of visitors who contribute in pool. On the other hand, the visitors do not like to reveal their vote to the administrator for privacy purpose. In this case, each user sends an integer value instead of the Boolean value and the range of V is different from the number of users. For example, in mentioned sample the range of V is 5 × Number of users. So there are two main parameters that should be considered: Number of the visitors and the Range of V. In this case, the time is highly related to the range of V. The algorithm is tested over a range of 100 to 1 million users and a range of V between 100 to 100 million. The result is shown in Fig.2.b.

6. Conclusions

Although, the processing time is highly related to the number of visitors, but the range of *V* should not be ignored. The result shows that by increasing the range of *V* the final processing time extremely increased. It is due to the complexity of the Teller algorithm's loop; because the loop variable is *V*. If the number of user increases, the time of computing *m*, *h and* shared keys will increase. Note that two shared secret keys Ω, Ψ are also furthered. Even though, this mutation is considerable, but the range of *V* and its role in try and error loop is more significant. The algorithm lasts less than 40 minutes for 1 million voters with *V* in the range of 1 to 100 million.

Using the infinite convergent product series has two main benefits; first the shard secret key is independent of the participants' keys so the process can continue online during application life and second the infinite feature of the product sequences provides the possibility of the infinite number of participants who like to join the process.

7. References

[1] Bogetoft Peter, Christensen Dan Lund, Damg Ivan, rd, Geisler Martin, Jakobsen Thomas, Kr Mikkel, igaard, Nielsen Janus Dam, Nielsen Jesper Buus, Nielsen Kurt, Pagter Jakob, Schwartzbach Michael, and Toft Tomas, "Secure Multiparty Computation Goes Live", in Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers (Springer-Verlag, 2009), pp. 325-343.

[2] O. Goldreich, "Foundations of cryptography: a primer Found," Trends Theor. Comput. Sci. 1 (1), 1-116 (2005).

[3] W. Du and Z. Zhan, "A practical approach to solve Secure Multi-party Computation problems", in workshop on New security paradigms (2002), pp. 127-135.

[4] Christian Schaffner, "Simple protocols for oblivious transfer and secure identification in the noisy-quantum-storage model," Physical Review A 82 (3), 032308 (2010).

[5] M. Naor and B. Pinkas, "Distributed Oblivious Transfer," Advances in Cryptology -- Asiacrypt '00, LNCS 1976, Springer-Verlag, 200-219 (December 2000).

[6] Sven Laur and Helger Lipmaa, "On Security of Sublinear Oblivious Transfer," Draft (2006).

[7] Kok Seng Wong and Myung Ho Kim, "Semi-trusted collaborative framework for multi-party computation," KSII Transactions on Internet and Information Systems 4 (3) (2010).

[8] M. Naor and B. Pinkas, "Efficient Oblivious Transfer Protocols", in SODA 2001 (SIAM Symposium on Discrete Algorithms) (Washington DC, 2001).

[9] Ivan Damgård and Jesper Nielsen, "Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption", in In Advances in Cryptology - CRYPTO'03 (2003), Vol. 2729, pp. 247-264.

[10] A. Shamir, "How to share a secret," Communications of the ACM, 22(11) 612–613 (1979).

[11] C. Blundo and B. Masucci, "Randomness in Multi-Secret Sharing Schemes," Journal of Universal Computer Science 5 (7), 367--389 (1999).

[12] Rosario Gennaro, Michael O. Rabin, and Tal Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography", in Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing (ACM, Puerto Vallarta, Mexico, 1998), pp. 101-111.

[13] Qin Li, W. H. Chan, and Dong-Yang Long, "Semiquantum secret sharing using entangled states," Physical Review A 82 (2), 022303 (2010).

[14] Abdolreza Rasouli, Javad Hosseinkhani, Mahboubeh Shamsi, and Majid Harouni, "A Robust and High Speed E-Voting Algorithm Using ElGammel CryptoSystem", in The 2nd International Conference on Computer and Automation Engineering (Singapore, 2010).

[15] Abdolreza Rasouli, Mohd Aizaini Bin Maarof, and Mahboubeh Shamsi, "A New Multi Party Aggregation Algorithm Using Infinite Product Series", in The 9th WSEAS International Conference on Applications of Computer Engineering (Penang, Malaysia, 2010).

8. References (This is "Header 1" style)

(Book style)

[1] A. Gray. *Modern Differential Geometry*. Publication place: CRE Press, 1998.

(Book style with paper title and editor)

[2] J. Li, Y. Jiang, and R. Fan. Recognition of Biological Signal Mixed Based on Wavelet Analysis. In: Y. Jiang, *et al.* (ed.). *Proc. of UK-China Sports Engineering Workshop*. Liverpool: World Academic Union. 2007, pp. 1-8. (Use "References" Style)

[3] R. A. Scholtz. The Spread Spectrum Concept. In N. Abramson (ed.). *Multiple Access*. Piscataway, NJ: IEEE Press, 1993, ch. 3, pp. 121-123.

(Journal style)

[4] R. Dewri and N. Chakraborti. Simulating recrystallization through cellular automata and genetic algorithms. *Modelling Simul. Mater. Sci. Eng*. 2005, 13(3): 173-183.

[5] S. Chen, B. Mulgrew, and P. M. Grant. A clustering technique for digital communications channel equalization using radial basis function networks. *IEEE Trans. on Neural Networks*. July 1993, 4: 570-578.

(Published Conference Proceedings style)

[6] S. P. Bingulac. On the compatibility of adaptive controllers. *Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory*. New York, 1994: 8-16.

(Presented Conference Paper style)

[7] G. W. Juette and L. E. Zeffanella. Radio noise currents n short sections on bundle conductors. Presented at the IEEE Summer Power Meeting. Dallas, TX, June 22-27, 1990.

(Patent style)

[8] J. P. Wilkinson. Nonlinear resonant circuit devices. U.S. Patent 3 624 12. July 16, 1990.

(Standards style)

[9] *Letter Symbols for Quantities*. ANSI Standard Y10.5-1968.

(Handbook style)

[10] *Transmission Systems for Communications*. 3rd ed. Western Electric Co., Winston-Salem, NC, 1985: pp. 44-60.

[11] *Motorola Semiconductor Data Manual*. Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.