Secure Sharing Sensitive Data Based on Network Coding and Attribute-Based Encryption

Zhiqiang Xu[1(✉)], Bo Shen[2], and Zhiyuan Zhang[2]

[1] School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
18120155@bjtu.edu.com

[2] Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing, China
{bshen,zhangzhiyuan}@bjtu.edu.com

Abstract. The security of sharing sensitive information through a distributed information storage and sharing platform is required strictly in many situations. In this paper, we present a novel approach with the aim of increasing security of sharing sensitive information. The model improves security by exploiting Attribute-Based Encryption. In addition, we incorporate network coding to our model to improve the efficiency of transmitting information.

Keywords: Distributed information storage · Attribute-based encryption · Network coding

1 Introduction

With the development of information technology, people can quickly obtain valuable information from various types of data sources nowadays. We can easily facilitate cross-domain data sharing by a lot of means, such as Internet, massively parallel processing databases, distributed file systems, cloud computing platforms and scalable storage systems. Information acquisition and sharing is the fundamental purpose of the development of these communication and network technologies, and it is also an important technical means to build various information systems. At the same time, the distributed, cross-domain and dynamic transmission and sharing of information are also facing great security threats, which put great challenges to the traditional security model.

The security of sharing sensitive information is through a distributed information storage and sharing platform. Under the premise of ensuring information security, the sharing provider can share the required sensitive information according to a certain strategy, scope and target.

For the purpose of security, access control and data management are realized through security isolation between multiple domains in general information systems. On the other hand, cross-domain access is often required to maximize shared network and data resource services which are contradict to each other. Because cross-domain access

Y.-B. Lin and D.-J. Deng (Eds.): SGIoT 2020, LNICST 354, pp. 95–105, 2021.
https://doi.org/10.1007/978-3-030-69514-9_9

makes the management of users with different security domains and different security levels more and more complex, and various application layer protocol vulnerabilities give hackers opportunity to attack the network. Therefore, it is very important to study the security sharing theory and technology of sensitive data in view of the application scenario of multi-level security cross-domain sharing of data, and realize flexible cross-domain data access and improve the availability of data under the condition of protecting sensitive data from leakage.

In this paper, we describe a secure approach to share sensitive information with network coding and Attribute-Based Encryption in order to leverage the efficiency and security of information transmission.

2 Related Work

Shamir [1] first proposed the concept of Identity-Based Encryption. They introduce a novel cryptographic scheme, which enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. However, it wasn't until much later that Boneh and Franklin [2] proposed the first fully functional Encryption scheme that was both practical and secure. Their solution made novel use of groups for which there was an efficiently computable bilinear map. The scheme had chosen ciphertext security in the random oracle model assuming an elliptic curve variant of the computational Diffie-Hellman problem.

Canetti et al. [3] rigorously defined a notion of security for forward-secure public-key encryption and also gave efficient constructions of schemes satisfying this notion. They proved semantic security of one scheme in the standard model based on the decisional version of the bilinear Diffie-Hellman assumption. Yao et al. [4] presented a scalable and joining-time-oblivious forward-secure hierarchical identity-based encryption scheme that allows keys to be updated autonomously. They also noted how their techniques for resisting collusion attacks were useful in attribute-based encryption. However, the cost of their scheme in terms of computation, private key size, and ciphertext size increased exponentially with the number of attributes.

Sahai and Waters [5] propose a new type of Identity-Based Encryption called Fuzzy Identity-Based Encryption as a new means for encrypted access control. In a Fuzzy Identity-Based encryption system ciphertexts are not necessarily encrypted to one particular user as in traditional public key cryptography. Instead both users' private keys and ciphertexts will be associated with a set of descriptive attributes or a policy over attributes. A user is able to decrypt a ciphertext if there is a "match" between his private key and the ciphertext.

Bethencourt et al. [6] provided the first construction of a ciphertext-policy attribute-based encryption (CP-ABE), and gave the first construction of such a scheme. In their system, a user's private key would be associated with an arbitrary number of attributes expressed as strings. On the other hand, when a party encrypted a message in the system, they specified an associated access structure over attributes. A user would only be able to decrypt a cipher-text if that user's attributes pass through the cipher-text's access structure. At a mathematical level, access structures in their system were described by a

monotonic "access tree", where nodes of the access structure were composed of threshold gates and the leaves describe attributes. They noted that AND gates could be constructed as n-of-n threshold gates and OR gates as 1-of-n threshold gates. Furthermore, they could handle more complex access controls such as numeric ranges by converting them to small access trees.

Ahlswede et al. [7] proposed a new class of problems called network information flow which is inspired by computer network applications. Consider a point-to-point communication network on which a number of information sources are to be multicast to certain sets of destinations. We assume that the information sources are mutually independent. The problem is to characterize the admissible coding rate region. This model subsumes all previously studied models along the same line. In this paper, we study the problem with one information source, and we have obtained a simple characterization of the admissible coding rate region. Our result can be regarded as the Max-flow Min-cut Theorem for network information flow. Contrary to one's intuition, our work reveals that it is in general not optimal to regard the information to be multicast as a "fluid" which can simply be routed or replicated. Rather, by employing coding at the nodes, which we refer to as network coding, bandwidth can in general be saved. This finding may have significant impact on future design of switching systems.

3 Methodology

3.1 CP-ABE

First of all, we give the definitions of an access structure proposed by Bethencourt et al. [6]. Then we introduce how does the ciphertext-policy attribute-based encryption scheme work to guarantee the security of access control.

Definition (Access Structure): Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$ if$B \in \mathbb{A}$ *and* $B \subseteq C$ *then* $C \subseteq \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) A of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e.,$\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$\empty set. The sets in A are called the authorized sets, and the sets no in A are called the unauthorized sets.

Setup. The input of setup algorithm is the implicit security parameter and the output are public parameters PK and a master key MK.

Encrypt(PK, M, A). The input of encryption algorithm are the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt the message M and output a ciphertext CT which contains A implicitly. The ciphertext CT can only be decrypted rightly when a user have a set of attributes that satisfy the access structure A.

Key Generation(MK, S). The input of key generation algorithm are the master key MK and a set of attributes S which gives some descriptions of the key. And the output of this algorithm is a private key SK which contains the set of attributes S implicitly.

Decrypt(PK, CT, SK). The input of decryption algorithm are the public parameters PK, a ciphertext CT, and a private key SK. As discussed in the former algorithms, we can see that the ciphertext CT contains access structure A implicitly and the private key SK contains the set of attributes S implicitly. When the set S of attributes satisfies the access structure A, the decrypt algorithm will decrypt the ciphertext rightly and output a message M.

Delegate(SK, S'). The input of delegate algorithm is a secret key SK of set of attributes S and a set S'⊆S. the output of this algorithm is a secret key S'K for the set of attributes S'.

There are several steps when we want to use CP-ABE. First of all, we need a implicit security parameter to generate the public parameters PK and a master key MK. Secondly, when we want to encrypt a message M, Encrypt(PK, M, A) can output a ciphertext CT. Then we should use Key Generation(MK,S) to generate private key SK for every users. Finally, user can use his private key SK to get the message M through the function Decrypt(PK, CT, SK) if the SK satisfies the access structure A.

3.2 Network Coding

Network coding is a novel technique proposed in 2000 which can improve throughput and performance of network. Many people think it will be a critical technology for future networks. With the appearance of network coding, a simple but important observation was made that in communication networks, we can allow nodes to process as well as forward the incoming independent information flows. When an intermediate node gets some independent data, it can make binary addition before transport them. Handling these independent data streams can greatly improve the efficiency of network information transmission. It is the cheap computational power that network coding utilizes to greatly increase network throughput. We will give a simple example to show how does the network coding improve network throughput when multicasting.

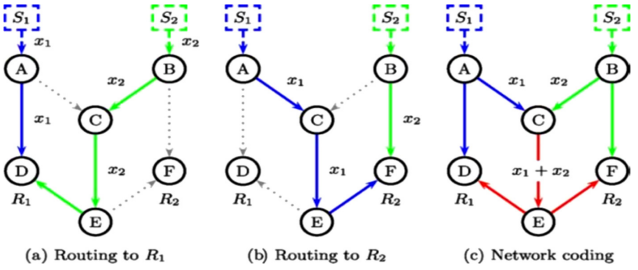

Fig. 1. Sources S1 and S2 multicast data to receivers R1 and R2.

As we can see in Fig. 1, a communication network represented as a directed graph, vertices represent terminals and edges represent channels. This communication network is commonly known as the butterfly network in the network coding. Assume that we have

slotted time, and that through each channel we can send one bit per time slot. We have two data sources S1 and S2, and two data receivers R1 and R2. Each source produces one bit per time slot. Data x1 is produced by S1 and x2 is produced by S2.

If receiver R1 uses all the network resources by itself, it could receive both sources. As shown in Fig. 1(a), we can route the bit x1 from source S1 through the path {AD} and the bit x2 from source S2 through the path {BC, CE, ED}. In the same way, if we want the second receiver R2 with using all the network resources by itself, it could also receive both sources. As shown in Fig. 1(b), we can route the bit x1 from source S1 through the path {AC, CE, EF}, and the bit x2 from source S2 through the path {BF}.

We consider a situation called multicasting that receivers R1 and R2 want to receive the information from sources S1 and S2 at the same time. When comes to this situation, receiver R1 can receive x1 from source S1 through the path {AD} and receiver R2 can receive x2 from source S2 through the path {BF}. But we want receiver R1 can get x2 from source S2 and receiver R2 can get x1 from source S1 through the path {CE} simultaneously. However, we can only send one bit per time slot and we want to simultaneously send bit x1 to reach receiver R2 and bit x2 to reach receiver R1.

In the past, information flow was treated like fluid through pipes, and independent information flows were kept separate. In this case we would have to make a decision at edge CE: either use it to send bit x1, or use it to send bit x2. When we decide to send bit x2 using edge CE, then receiver R2 will only receive x2, at the same time receiver R1 will receive both x1 and x2.

The simple but important observation made by Ahlswede et al. [7] is that we can allow intermediate nodes in the network to process their incoming information streams instead of just forward them. In particular, node C can take bits x1 and x2 and xor them to create a third bit $x3 = x1 + x2$ which it can then send through edge CE (the xor operation corresponds to addition over the binary field). R1 receives $\{x1, x1 + x2\}$, and can solve this system of equations to retrieve x1 and x2. Similarly, R2 receives $\{x2, x1 + x2\}$, and can solve this system of equations to retrieve x1 and x2.

This example shows that when multicasting we can improve throughput through allowing intermediate node in the network to process information streams and receivers exacting the information.

In addition to the improvement of throughput, the security of network coding also has advantages. Instead of sending uncoded data, sending linear combinations of data can offer a natural way to use advantage of multipath diversity for security against wiretapping attacks. So we only need protection against other attacks with network coding.

However, there are some challenges to deploy the network coding. First of all, the complexity of employ network coding is high because the nodes in network need to have some additional functionalities. As we discussed in Fig. 1, the intermediate nodes C has additional memory and calculation requirements because instead of broadcasting streams immediately, it needs to store and process streams. The receivers D and F should be able to exact the information through what it receives. In fact, it is difficult to access the complexity of network coding. Secondly, with the rapid development of network technology, our demand for network security is becoming higher and higher. In our file sharing scheme, security is an important part and we need to guarantee protection against

sophisticated attacks. Network coding need intermediate nodes to perform operations on the data. Thus, we should prevent intermediate nodes from maliciously tampering with the data. Finally, as communication networks' high speed development, a challenging task is to incorporate the emerging technologies such as network coding, into the existing network architecture. We want to profit from the network coding without incurring dramatic changes in the existing equipment and software. A related open question is, how could network coding be integrated in current networking protocols.

3.3 Model Description

We describe our model in this section. In our simulated file transfer system, we have one controller server and seven file servers, and the connection topology is shown in the Fig. 2. Each file server is connected to the controller server. These file servers are server nodes in different networks, representing system nodes which are located in different areas. Every server node stores a variety of encrypted files. In the initialization phase of the system, all the file server nodes first establish a connection to the controller and then transmit some information to it. The information is about all the local file directories and the connections to other file server nodes. When the controller receives feedback information from each node, it will summarize information. Through connections of each node, the controller can get the whole file system nodes' connection diagram. In the same way, after receiving every node's local file directory, the controller can generate a list of all the files existing in the system which consists of two parts. The first part is a map of in which nodes each file locates, the second is the directory of every node. In order to make our system has the ability of resisting some disasters, we follow the idea of Hadoop's mechanism which is making two copies of all files, that is to say for each file, has two backups distributed in different nodes of the system. This approach has great benefits for the system. Image a scene, if some special circumstances such as a node is damaged happen suddenly, we can still use the backup files stored in other nodes for file sharing and the recovery of damaged node. To achieve this, after the initialization process, system will start to maintain the list of files regularly. During list's maintenance, the controller sends a request to each server node. After receiving the request, server node will return the hash values of all the local files to the controller in the form of (file name, 32-bit md5 value). When received the files' hash values returned by each node, the controller compares the 32-bit md5 values of the same file, and there will be three different situations. In the first case, three backups of the same file have the same md5 value, which means that each file is right and no additional operations are required. The second case, two backups' md5 values are same, but are different from the last ones. Two files of same md5 value are right and the file which it's md5 value is different from the other two backups is damaged. The controller will notify the corresponding node stored damaged file to delete the damaged filed. Then it will randomly select a node which has same backup without error to transmit this file to the error node. Then we can ensure that there are two right backups of each file in our system all the time; Last case is that three backups' md5 value are different from each other which means at least two files are destroyed. We have no idea to distinguish which one is right. In this situation, the controller will notify the three server nodes to delete this file after which system will update the list of all files. But in fact the last kind of circumstance rarely happens.

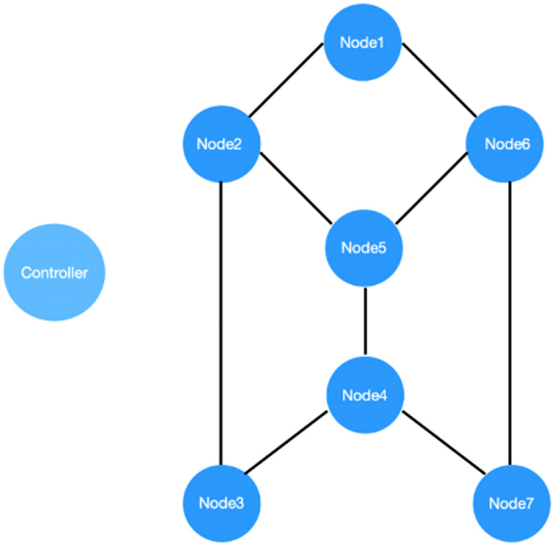

Fig. 2. Connection topology of simulated file transfer system

Another important task in the system's initialization phase is to assign private keys of ABE to each server node. We need to be aware of every server node's attributes due to use of attribute-based encryption in our system. We set default property set as the form of (node_id: id). For example, node 3 has attribute as a string of "node_id: 3". After determined attributes for each node, the controller generates private keys with the corresponding attributes of each node through key generation algorithm in ABE and then sends private keys to related server nodes.

When the controller assigns the private keys to server nodes, it is not allowed to send plaintext directly. Our system uses SSL (Secure Socket Layer) to distribute private keys. SSL is a protocol for the secure exchange of information between TCP connections which provides two basic security services: authentication and confidentiality. To use SSL for messaging, some preparations are needed on the controller side like installing openssl. First of all, an application for root certificate is generated using the private key, and root certificate will be produced using the self-signed method to sign the previous application. Then the private key of the server's authentication certificate is established. We are supposed to generate the certificate application file, and the root certificate is used to issue the server's authentication certificate. Finally, the private key of client's authentication certificate is established and certificate application is generated. Client's authentication certificate is issued by root certificate. After generating all the certificates, we assign the server-side certificate to the controller and the clients' certificates to the various file server nodes in the system. We can safely send the private keys of ABE to each file server node through the controller with SSL methods after finish preparatory work.

We will briefly make a explanation of ABE's encryption and decryption strategy. Suppose our attribute set is {student, teacher, male, female, school of electronic and

information engineering, school of economics and management, school of science}. The first step is choosing some attributes from the set to form the policy of encryption and decryption and the nodes which match chosen attributes can decrypt ciphertext correctly. For example, if the policy is selected as {teacher, male}, as long as the user's attributes contains "teacher" and "male", it can decrypt ciphertext correctly. If there is a user whose attributes are {school of electronic and information engineering, teacher, male}, he can decrypt correctly because his attributes match the policy. If another user's attributes are {school of electronic and information engineering, teacher, female} then she cannot properly decrypt because her attributes do not satisfy attributes described in the policy. Through this simple example, we can find out that in the ABE encryption scheme, the private key of each user will do not change after generated. It is the policy that changes all the time to satisfy different demands of encryption and decryption.

After the initialization of the system is completed, we can start file sharing. If user needs a file, he needs to issue a request contains the desired file's name to the closest file server node to which he connects. When a server node receives request of file, first it will research that file locally. If server node has this file, it sends to the user directly. If this file is not available locally, the server node will send file request to the controller in the form of desired file's name. After receives a file request, the controller will search through the list of all files according to desired file's name to find where it is stored. When it comes to multicast, first of all the controller calls the minimum cut maximum flow algorithm of network coding to find out the path of transmission. Then the weight matrix is calculated by the weight calculation function in the network coding. In the following time, the configuration information will be sent to corresponding server nodes according to the path. After receiving configuration information, these nodes will prepare to receive and forward files as required. Controller can choose encryption policy according to attributes of user who requests files. The policy will be sent to first server node of the path. Starting node divide the file into some parts in setting size (for example, we set the block size of 100 MB). Every part will be encrypted by ABE with a header which recording the location of each part. UDP packet is used in the sending process, but only after receiving the confirmed message from receiver, the sending side can continue to send the next packet. Moreover, we have set up a simple verification and retransmission mechanism to ensure the high transmission quality of files. When get some bitstreams, receiver puts it into a queue and enables another thread to process bitstreams according to the configuration information of network coding and forwards it. Sending and receiving are separated. After receiving all the segmented parts, end server node extract information with network coding algorithm. Then it will decrypt segmented part using private key and recover complete file according to the header. This is what happens when files are transmitted in our system of multicast. If it is unicast, the controller finds the path according to the shortest path algorithm, and then sends the configuration information to related nodes. When server node is prepared well, the encryption policy is selected and sent to the starting node of path. The starting server node divides the file into some parts, encrypts these with ABE according to the policy, and then transmits the file. The intermediate node forwards the received bitstreams directly without any processing. After receiving each file part, the end server node will finish decryption and merge them into a complete file according to the header.

4 Experiment

We describe the evaluation tasks and report the experimental results in this section. We evaluate ABAE on two criteria:

1. Security
2. Effective of network coding

The system security of this paper mainly consists of two parts: the security of private key distribution and the security of file sharing. For key distribution, SSL is used to transmit private key securely because of its bidirectional authentication function. The related certificates are distributed to the controller node and the server nodes so that they can authenticate each other, and their communications will be encrypted when the authentication is completed. We use network sniffer tool to monitor the traffic which is found that all the data transmitted is garbled in the process of private key distribution. Then we can guarantee the security of key distribution process.

In addition to the security of key distribution, it is also necessary to ensure the security in process of sharing files. We use ABE to encrypt the transmitted content, because as required in distributed system, the data can be accessed correctly only if the user satisfies some particular attributes. However, there is a risk that the server node will be attacked if files are stored on it. Then we use Ciphertext-Policy Attribute-Based Encryption to ensure access control. The decryption policy is under the control of the controller node, so that even if the server where files are stored is attacked, data can be protected. [Ciphertext-Policy Attribute-Based Encryption] has proved that CP-ABE can resist collusion attacks.

We want to verify the efficiency of network coding in multicast. In this regard, we respectively transfer two different files from node 1 to node 3 and node 7 to record the speed. Each node stores 50 to 100 files, ranging in size from 1 MB to 5000 MB. We change size of the transferred files and recorded the time required for transferring two files under the two methods of using network coding and not using network encoding for comparison. The results are shown in the Table 1 when two files are in same size. The results are shown in the Table 2 when one file is half the size of the another.

Table 1. The time of transferring files (two files are in same size) using and not using network coding

	The size of files (MB)							
	1	5	10	50	100	500	1000	5000
With network coding (ms)	603	2912	5863	23829	52952	246922	508574	2297855
Without network coding (ms)	605	3115	6322	27176	57835	275357	565632	2582117

Table 2. The time of transferring files (one file is half the size of the another) using and not using network coding

	The size of bigger file (MB)							
	1	5	10	50	100	500	1000	5000
With network coding (ms)	449	2177	4510	18459	40998	193818	400105	1837972
Without network coding (ms)	457	2298	4801	20082	44019	206509	424324	1948675

As we can see in the experiment, the transferable efficiency of using network coding is higher than not using it by almost 11% when two files are in same size. But when one file is half the size of another the efficiency improvement is decrease to almost 6% because of the limitation of network coding which is that it is most efficient to transfer files of the same size. We can get the conclusion that using network coding can improve transferable efficiency in our system.

5 Conclusion

In this paper, we proposed a system for share sensitive data securely which was based on attribute-based encryption and network coding. Our system uses a new type of encrypted access control where user's private keys are specified by a set of attributes and a party encrypting data can specify a policy over these attributes specifying which users are able to decrypt. We also use network coding in our system to improve the efficiency of multicast.

One limitation of our system is that ABE is proved secure under the generic group heuristic. So the security of our system is limited in some situations. But we believe some more secure technologies which can be applied to our system would be proposed in future.

Acknowledgements. This work is supported by the National Key R&D Program of China under Grant 2018YFC0831300, China Postdoctoral Science Foundation under Grant 2018M641172, the National Natural Science Foundation of China under Grant 61701019.

References

1. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
2. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
3. Biham, E. (ed.): EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9

4. Yao, D., Fazio, N., Dodis, Y., et al.: ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In: ACM Conference on Computer & Communications Security. ACM (2004)
5. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, 2007. SP '07. IEEE (2007)
7. Ahlswede, R., Cai, N., Li, S.Y.R., et al.: Network information flow. IEEE Trans. Inf. Theory **46**(4), 1204–1216 (2000)
8. Fragouli, C., Soljanin, E.: Network coding fundamentals. Found. Trends Netw. **2**(1), 1–133 (2007)
9. Gamal, A.A.E., Cover, T.M.: Achievable rates for multiple descriptions. IEEE Trans. Inf. Theory **28**(6), 851–857 (1982)
10. Roche, J.R., Yeung, R.W., Hau, K.P.: Symmetrical multilevel diversity coding. IEEE Trans. Inf. Theory **43**(3), 1059–1064 (1997)
11. Yeung, R.W., Zhang, Z.: Distributed source coding for satellite communications. IEEE Trans. Inf. Theory **45**(4), 1111–1120 (1999)
12. Agarwal, A., Charikar, M.: Manufacturing engineer - on the advantage of network coding for improving network throughput. In: IEEE 2004 IEEE Information Theory Workshop - San Antonio, TX, USA (24–29 Oct. 2004), pp. 247–249 (2004)