The Security Aspects of the Cirrostratus Private Cloud Storage

Vitaly Petrov
Tampere University of Technology
Tampere, Finland
vitaly.petrov@tut.fi

Victor Minchenkov
State University of Aerospace Instrumentation
St. Petersburg, Russia
victor@vu.spb.ru

Stanislav Bogatyrev
EMC St. Petersburg Development Center
St, Petersburg, Russia
realloc@realloc.spb.ru

Abstract

The paper investigates the insecurities present in the Cirrostratus Private Cloud Storage. Vulnerabilities in the ATA over Ethernet (AoE) protocol, that is used for internal communication in the cloud, are enumerated. Possible attacks are analysed, and the protocol modification, resistant to the majority of them, is proposed. The security extension of the AoE protocol is illustrated by some code examples.

Index Terms: Network security, Cloud computing, ATA over Ethernet, AoE.

I. Introduction

Trends in the growths of the computer-communication networks seem to indicate that the worlds information double every two years, and in 2011 the world will create a staggering 1.8 zettabytes [1]. Therefore, there is a problem to provide a fast, secure, and, last but not least, cheap way to access the stored data. In this case, new storage architectures are proposed. And one of them is a ”cloud storage” — a model of data storage, where data is stored on multiple servers, but is accessible via the same way as remote disk.

From the security point of view, all cloud storage can be divided into two big groups: public and private. In the public cloud every user has the access to all the information stored in the cloud. Therefore, there is no need to encrypt the data, transmitted between servers in the cloud. On the contrary, in the private cloud all the security aspects, including confidentiality, integrity and accessibility should be considered.

The overwhelming majority of clouds, including Amazon [2] and Google [3], are online, and users access them through the Internet. This type of clouds has a lot of advantages. First, a minimum of changes in the company’s network are required. At second, such architecture is adjustable and can use more computer resources when needed on-the-fly, e. g. the last 3 days of the month. Moreover, all the routines (backup, software and hardware updates) are done automatically by the organisation that provides a cloud service. So, in general, for short-term usage Total Cost of Ownership (TCO) of such a cloud is lower, comparing with an internal one.

At the same time, internal cloud provides high enough scalability and could be more profitable for long-term projects. Moreover, there are cases, when it is better to store the company’s information inside the office. Mainly, they are associated with the security reasons and outside traffic limits. Therefore, here appears to be a very cheap (in order to compete with the online clouds), fast and secure cloud service. And one of the possible solutions is described below.

The approach is to propose a security extension for the private cloud storage, called Cirrostratus [4]. Due to the cloud developers demand, the solution should not change the existing frame format much and have to be easily implemented. Therefore, the proposed solution mostly uses libraries, that already exist in the Linux kernel. However, resistance to the major of possible attacks requires either. So the security extension provides confidentiality and integrity of each and any message. Unfortunatelly, as far as Cirrostratus production version is not released yet, no performance comparisson with the existing solutions could be done. This is one of the future research directions, as far as security level evaluation and improvement.

The paper has the following structure. In section II several data access protocols are compared. In section III the architecture of Cirrostratus is described. In section IV vulnerabilities of the chosen protocol and several attacks, based on it, are considered. In section V some improvements, providing confidentiality and integrity are proposed. And in section VI changes required in the AoE driver are listed as well as the modified protocol resistance to the most popular attacks. The paper ends with some conclusion remarks.

II. Linking data storage protocols survey

There are several protocols for communication between computing servers and disk drive devices. All of them work over Ethernet or TCP/IP layer. In this section only the most popular of them are described. There exist several good solutions based on Fibre Channel serial interface technology [6], but their implementation requires substantial investments into the office network infrastructure. This counters to the purpose of building cheap internal cloud. Therefore, such solutions are not included in the list.

A. Internet Small Computer System Interface (iSCSI)

iSCSI [5] is an IP-based protocol, that allows transmitting SCSI commands trough the existing IP network. It was designed for getting direct access to SCSI disks connected to the remote server (see fig. 1)

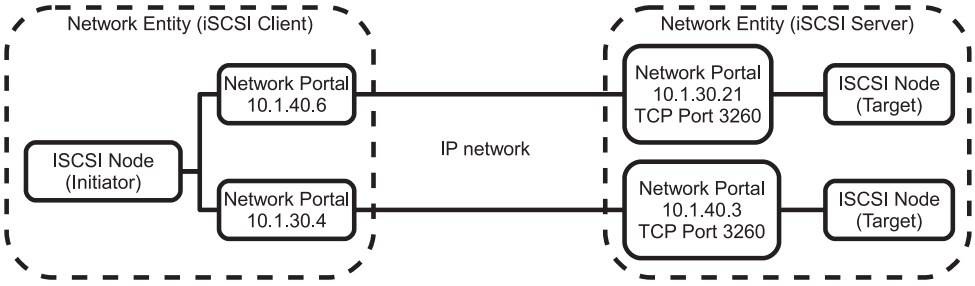

Fig. 1. iSCSI architecture model [5].

The benefits of iSCSI are obvious. Working over TCP/IP network, it provides routing and high-level reliability. Moreover, there are several authentication methods, including remote passwords and private keys. Therefore, the security level is sufficient for the protocol purpose. However, there are two considerable issues: high message redundancy and high complexity of the protocol. Issue amounts to a lower throughput is small and medium size networks and the second one — to the significant hardware price increase comparing with other standards.

B. HyperSCSI

Similar to iSCSI, HyperSCSI is used to bypass the SCSI commands through the network [7]. But instead of using TCP/IP, it works directly over the Ethernet layer. Therefore, it does not have issues, mentioned in the previous subsection. However, being proposed in 2002, HyperSCSI has not been adopted by commercial vendors yet. Moreover, in [8] it is shown that performance of the protocol is unstable with network congestion. So HyperSCSI cannot provide the required reliability level in high-load networks.

C. ATA over Ethernet (AoE)

ATA over Ethernet is a lightweight network protocol, designed for simple, high-performance access to SATA storage devices over the existing Ethernet network [9] . Being constructed to build low-cost storage area networks, it has a very simple header format (see fig. 2).

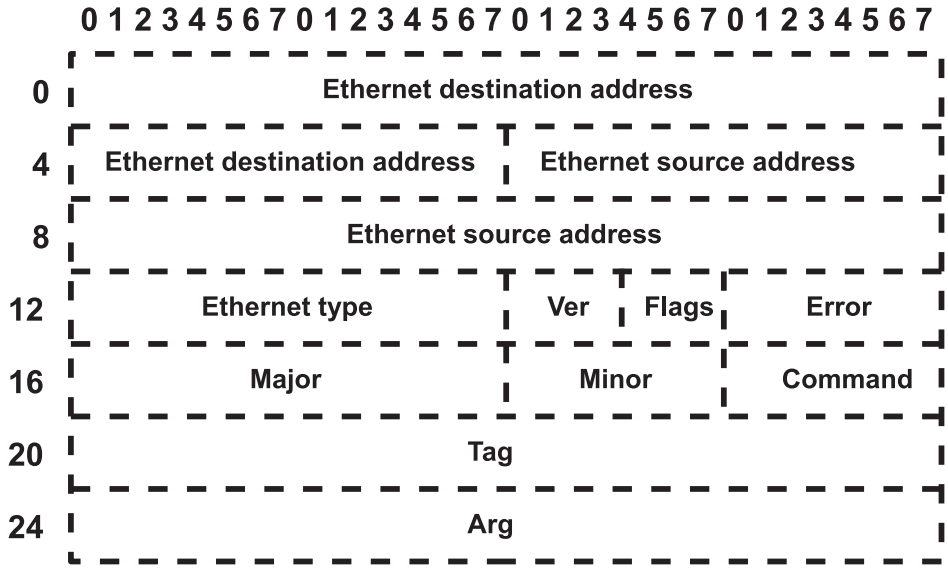

Fig. 2. AoE header format [9].

Because of this, AoE provides very low message redundancy level in small storage area networks. In addition, because of standard simplicity, AoE is easy to implement in both software and hardware. So it does not require huge investments into the network infrastructure. Moreover, Linux kernel from version 2.6.5 supports AoE without any additional modifications. Therefore, AoE fits all the requirements to build a cheap internal cloud. However, it should be mentioned, that AoE is not a connection-based protocol. So each message should be consider as unique and unreliable.

III. CIRROSTRATUS ARCHITECTURE

The detailed description of the Cirroustratus can be found in [4]. Here only the primary aspects, that are important for talking about the security, are considered. All cirrostratus servers can be divided into two big groups: Computing Nodes (CN) and Storage Nodes (SN). The purpose of computing node is to hold several virtual machines on it and provide services to the end-user. Therefore, such nodes should have a high-performance CPU and a lot of RAM. In order to balance the workload, there should be the possibility to start each virtual machine on any computing node. So Virtual Hard Drives (VHD) need to be saved outside

the CN node. Moreover, all the CN nodes should have a fast connection to the virtual hard drive storage. Therefore, the workload need to be balanced again. One of the possible ways is to divide the VHD image into small blocks and distribute them between several storage nodes. But standard data access protocols are not able to work with files scattered to many nodes. Therefore, some middle ware layer should appear. Let us call these middleware servers Access Nodes (AN). Access node is a storage node that not only stores parts of VHD images, but also provides the access to other storage nodes (see fig. 3).

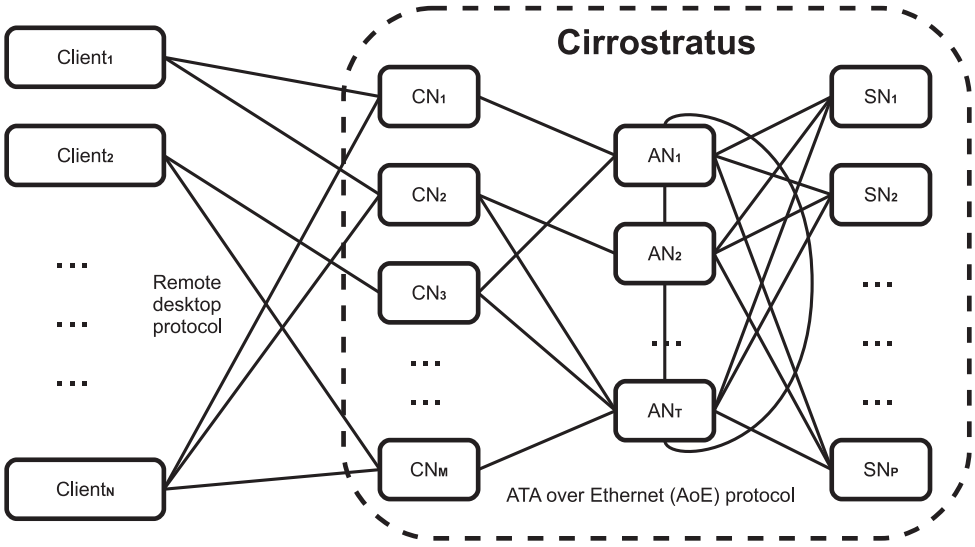

Fig. 3. Cirrostratus architecture.

When a computing node wants to access the particular VHD image, it initiates a connection with an access node via AoE protocol. AN, pretending to have the whole VHD on it, sends the request to the SN, holding the required VHD segment(s). Information about the VHD segments and SN, storing them, is calculated using current network map. The same modified CRUSH algorithm function is used for both scattering and gathering parts. If there is more than 1 access node in the network, the CN initially sends request to all of them and ask for the list of ”stored” VHDs. The image division and distribution between the storage nodes and reliability aspects are described in [10].

IV. Security issues in the AoE protocol

If all the CN, AN and SN are trusted nodes and situated in the dedicated network segment, there is no need for any security extensions. But, unfortunately, to reduce the investments in the network hardware, in small offices all the computers are usually connected to a single switch. So the real cirrostratus structure is a bit different to the described above (see fig. 4), because all the nodes use the single channel.

In this case, any computer in the office network can pretend to be a CN, AN or a SN. As far as the basic AoE protocol does not consider any security aspects, there are several possible attacks in such a network. The detailed description of most part of them can be found in [11].

1) *Replay attack*. As it is shown in the fig. 2, AoE packets has no sequence number field. Therefore, AoE packet can be simply captured and retransmitted after a while. This kind of attack can be used for rewriting changed data with an old value.

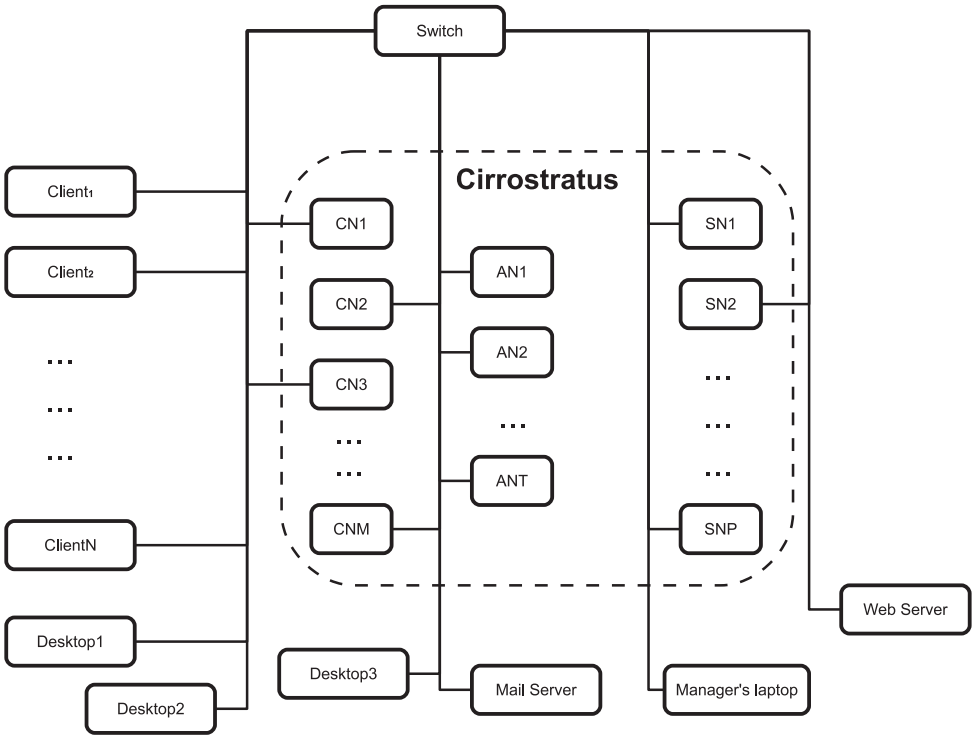

Fig. 4. The real cirrostratus architecture.

2) *Unauthenticated disk access attack*. As soon as the only authentication procedure in the original AoE protocol is MAC-address filtering and MAC can be easily spoofed, there is no way to authenticate the node. Therefore, the following attack is possible. Attacker chooses the node, where there is data he is interested in, and start sniffing all the AoE traffic destined to this node. Using the source MAC from the cached packets, he can form the AoE request with a special command and get the direct access to the drive. However, it is important to consider, that Attacker writes *directly* to the disk, without the knowledge of the cache, file system, etc... So he can easily disclose himself, if the system crashes.
3) *Man-in-the-middle attack*. By ARP-spoofing attack or similar actions, Attacker can modify the MAC-to-port routing table in the switch and let all the traffic between two hosts come through his own computer.
4) *DOS attack*. This attack can be simply done by both of the Reply attack and Unauthenticated disk access attack. Attacker can start sending a huge number of AoE request with a fake MAC address to the victim node. According to the workload limits, the node can not process them, so the request queue length increases. And after a while all the AoE requests from the other nodes will be thrown off, because the queue is full — the server is not responding. Considering such type of attacks is not expecting to occur in the small size of the company network, Cirrostratus is aimed for, resistance to DOS attacks is not mandatory.

All mentioned attacks, except DOS, are possible because of the following vulnerabilities in the AoE protocol:

- absence of the sequence number;

- weak authentication procedure (MAC-address filtering);
- sending plaintext data.

In the next section the idea of protocol modification, that provides confidentiality and prevents most of mentioned attacks, is described.

V. AoE protocol modification

Before discussing the protocol modification, it is necessary to consider, that the overwhelming majority of existing network hardware support only fixed-size AoE packets. So, as soon as standard Ethernet frame size is limited to 1520 bytes, it is impossible to write 3 or more 512 bytes sectors to the remote disk by sending a single packet. Therefore, all data vectors sending to the disk should be divided into blocks of 1024 bytes (2 sectors). After adding an AoE header (24 bytes, see fig. 2), the maximum length of the packet comes to 1048 bytes. So $1520 - 1048 = 472$ bytes are still available and can be used for the security objects.

To provide confidentiality the encryption procedure is required. Unfortunately, according to the efficiency reasons, any public key crypto system is not capable here. Therefore, any pair of nodes should have their own key to encrypt the traffic. And only symmetric encryption can be used. So each node should have a key table similar to $O(n)$ size, where n is a number of nodes. Moreover, to reduce the computation time, it's better to use a cipher, that is already implemented in the Linux kernel. In the kernel 2.6 and higher several symmetric cyphers are implemented, but in this paper the AES-128 CBC [12] usage is proposed. It is very fast, the key length of 128 bytes is enough to resist the "guessing" attack [13], and since 2002 there was not found any serious drawbacks in the encryption procedure.

One more advantage is hardware implementation of this algorithm in late Intel and AMD processors has, that dramatically increases the encryption speed. The most serious problem of symmetric system usage is key distribution. However, considering the dedicated office case, there is a possibility to initially load them for the nodes, before the cloud starts.

The second primitive, that was added to the driver is hash function. And again, the computation time was the major metric. Therefore, the MD5 [14] algorithm was chosen, instead of more secure but complicated SHA-1 and SHA-2. Hash algorithm was used to sign the message for integrity reasons.

As far as the encryption key is the same for the set of messages, some unique identifier should be added to the packet in order to resist the "Replay attack" mentioned in the previous section. So the proposed packet structure is a bit different, comparing with the original one (see fig. 5 and 6).

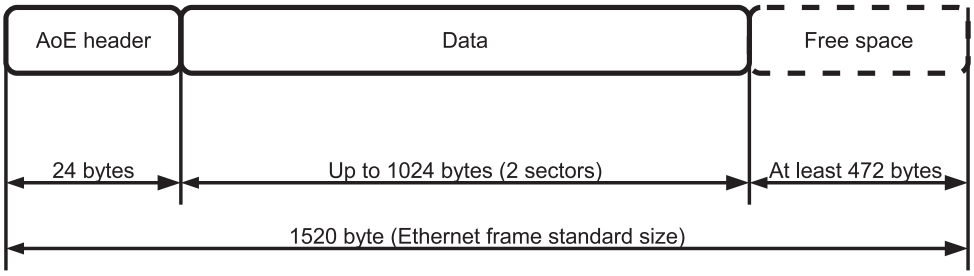

Fig. 5. The existing AoE packet structure.

So, before sending any data, Sender has to execute the following operations.

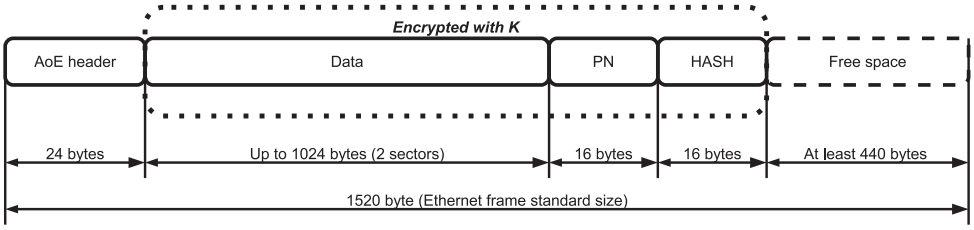

Fig. 6. The proposed AoE packet structure.

1) Add the current 4-bytes packet number ($PN_{ID_{Sender}}^{ID_{Receiver}}$) (from 0 to 65535) to the end of the data chunk.
2) Hash the data chunk together with the number using the MD5 algorithm and add the result to the end of the data chunk.
3) Encrypt the data chunk using the AES-128 algorithm and $K_{ID_{Sender}}^{ID_{Receiver}}$ key from the key table.
4) If the current $PN_{ID_{Sender}}^{ID_{Receiver}}$ is equal to 65535, then set it to 0 and update the encryption key using the MD5 hash function:

$$K_{ID_{Sender}}^{ID_{Receiver}} = hash(K_{ID_{Sender}}^{ID_{Receiver}})$$

If the current $PN_{ID_{Sender}}^{ID_{Receiver}}$ is lower than 65535, then increase it by 1.

At the same time, Receiver getting the message use the following algorithm to verify it.

1) Decrypt the data chunk using the AES-128 algorithm and the $K_{ID_{Sender}}^{ID_{Receiver}}$ key from the key table.
2) Check if the message PN is equal to current $PN_{ID_{Sender}}^{ID_{Receiver}}$, otherwise delete the message.
3) Hash the data chunk together with the PN using the MD5 algorithm and check if the calculated value equals the message hash sum, otherwise delete the message.
4) If the current $PN_{ID_{Sender}}^{ID_{Receiver}}$ is equal to 65535, then set it to 0 and update the encryption key using the MD5 hash function:

$$K_{ID_{Sender}}^{ID_{Receiver}} = hash(K_{ID_{Sender}}^{ID_{Receiver}})$$

If the current $PN_{ID_{Sender}}^{ID_{Receiver}}$ is lower than 65535, then increase it by 1.

If this algorithm is finished correctly, the incoming message is verified and can be sent to the upper network layer. The implementation of the security algorithms in the existing AoE Linux driver is described in the following section.

VI. Security mechanism implementation and analysis

Since 2.6 version the AoE driver is integrated into the Linux kernel. Therefore, only the C language and kernel API is allowed to use to modify it. According to algorithm, described in the previous section, at least 3 new functions should be added to the AoE driver: *encrypt*, *decrypt* and *hash*. Below there is their sketch listing on the C language.

```
static void encrypt(u8* dst, u8* src, u8* key)
{
        struct crypto_cipher* tfm;
        u32 type;
```

```
        u32 mask;
        unsigned int keylength;
        keylength = 16;
        type = 0;
        mask = 0;
        tfm = crypto_alloc_cipher("aes", type, mask);
        crypto_cipher_setkey(tfm, key, keylength);
        crypto_cipher_encrypt_one(tfm, dst, src);
        crypto_free_cipher(tfm);
}

static void decrypt(u8* dst, u8* src, u8* key)
{
        struct crypto_cipher* tfm;
        u32 type;
        u32 mask;
        unsigned int keylength;
        keylength = 16;
        type = 0;
        mask = CRYPTO_ALG_ASYNC;
        tfm = crypto_alloc_cipher("aes", type, mask);
        if(tfm == NULL)
        {
                fail();
        }
        crypto_cipher_setkey(tfm, key, keylength);
        crypto_cipher_decrypt_one(tfm, dst, src);
        crypto_free_cipher(tfm);
}

static void hash(u8* dst, u8* src, unsigned int src_len)
{
        struct crypto_hash *tfm;
        tfm = crypto_alloc_hash("md5", 0, CRYPTO_ALG_ASYNC);
        struct hash_desc desc = {.tfm = tfm};
        struct scatterlist sg[1];
        unsigned int ret;
        if(tfm == NULL)
        {
                fail();
        }
        ret = crypto_hash_init(&desc);
        if (ret != 0)
        {
            fail();
        }
```

```
        sg_init_table(sg, ARRAY_SIZE(sg));
        sg_set_buf (&sg[0], src, src_len); //
        ret = crypto_hash_digest(&desc, sg, ARRAY_SIZE(sg), dst);
        if (ret != 0)
        {
            fail();
        }
        crypto_free_hash(tfm);
}
```

Definitely, much more changes are required in the AoE driver to let it work property. And only the major aspects were discussed in this section. Some other implementation details can be found in [10]. But the printed listing illustrates the main idea of the protocol modification and the way they were implemented.

Now let us consider the modified protocol resistance to the attacks, described in section IV.

1) *Replay attack*. The proposed protocol is resistant to this attack, because each packet transmitted between two selected nodes has an unique number PN. Moreover, if this number is modified by anybody, who does not know the encryption key, the hash sum changes and Receiver will notice it during the verification procedure (see section V).
2) *Unauthenticated disk access attack* and *Man-in-the-middle attack*. The only other node, who has the $PN_{ID_{Sender}}^{ID_{Receiver}}$ key, except Receiver, is Sender. Therefore, he is the only person, who is able to send the appropriate AoE packet and pass the verification procedure. So the proposed protocol is resistant to the unauthenticated disk access attack as well as the Man-in-the-middle attack
3) *DOS attack*. Unfortunately, the modified protocol is not resistant to the DOS attack. Moreover, it is even more vulnerable to it, because of increased computation complexity during the verification procedure. However, in the small and medium size local networks — the main Cirrostratus market — such kind of attacks can easily be detected by the rapidly increased amount of traffic.

Despite the fact, that the modified protocol is resistant to the major part of mentioned attacks, there is at least one more serious drawback in the proposed solution. Considering 802.3 Ethernet cable network to be much more reliable, comparing with wireless alternatives (e. g. 802.11 and 802.16), we have to admit that even in the small network exists a possibility of losing a single packet. Unfortunately, AoE protocol does not provide any mechanism of such a mess detection. Therefore, there is a small probability, that request will not reach the Receiver. This packet loss, caused by noise or bad connection, can come for both original and modified protocols to a situation when the new data were not written to the disk. So far the whole system can become unstable. But for the modified protocol it comes, moreover, to the case, when the packet number counters PN of Sender and Receiver mismatch. So the next set of packets (before the encrypting key changes) will not be accepted. In the real network the probability of losing the Ethernet packet is almost 0, but talking about reliable storage services, even this is not enough. So some further research should be done to get rid of such a drawback.

Except this, the presented research is going to be extended in at least two directions. First of all, the performance of the proposed solution should be evaluated on the real hardware and compared with similar solutions using different protocols (e. g. iSCSI [5]) or online

clouds (such as Amazon [2] and Dropbox [15]). One of the possible methods is described in [16].Moreover, the security level can be raised by replacing the MD5 algorithm that has several flaws [17] with more secured but also more complicated alteratives, such as SHA. In addition, the default encryption mode can be changed from CBC to mordern XTS [18]. This will double the keys table size, but make the protocol resistant to known encryption mode attacks [19].

VII. Conclusions

Cloud computing represents a new paradigm in the data storage and processing. Despite there are several good online services, they should be replaced by internal solutions in some special cases. These solutions have to base on commodity hardware and use fast, reliable and secure protocols for communication between nodes. In this paper the security aspects of one of such protocols — ATA over Ethernet (AoE) — are considered. Favouring simplicity, mentioned protocol does not provide any encryption or serious authentication mechanisms. So both confidentiality and integrity of the message could be easily broken. After the typical network structure description and protocol vulnerabilities enumeration, finally, the protocol modification, resistant to the majority of possible attacks is proposed.

References

[1] EMC Corporation, "EMC digital universe study," http://www.emc.com/leadership/programs/digital-universe.htm.
[2] Amazon Inc., "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/
[3] Google Inc., "Google Cloud Connect for Microsoft Office," http://www.google.com/apps/intl/en/business/officeconnect.html
[4] S. Bogatyrev, "Cirrostratus first overview," http://www.slideshare.net/realloc/cirrostratus-first-overview
[5] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)," *RFC 3720*, 2004
[6] Preston, W. Curtis, "Fibre Channel Architecture," *Using SANs and NAS. Sebastopol, CA: O'Reilly Media*, pp. 19-39, 2002.
[7] W. Y. H. Wang, H. N. Yeo, Y. L. Zhu, T. C. Chong, "Design and development of Ethernet-based storege area network protocol," *Proceedings of the 12th IEEE International Conference on Networks (IEEE)*, pp. 48-52, 2004.
[8] M. Gug, "Performance comparison between iSCSI and other hardware and software solutions," *Computing in High Energy and Nuclear Physics*, 2003.
[9] S. Hopkins, B. Coile, " ATA over Ethernet," http://support.coraid.com/documents/AoEr11.txt, 2009.
[10] S. Bogatyrev, "Distributed Storage System Cirrostratus," *7th All-Russian University conference for Young Scientists*, 2010.
[11] C. Purvis, "Access over Ethernet: Insecurities in AoE,", White paper, http://www.security-assessment.com/files/documents/whitepapers/AccessoverEthernet-InsecuritiesinAoE.pdf.
[12] J. Daemen, V. Rijmen, "The Design of Rijndael: AES — The Advanced Encryption Standard," *Springer*, 2002.
[13] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, R. L. Rivest, "Handbook of Applied Cryptography,", 1997.
[14] R. Rivest, "The MD5 Message-Digest Algorithm," *RFC 1321*, 1992.
[15] D. Houston, A. Ferdowsi, " Dropbox Web-based File Hosting Service," http://dropbox.com, 2008.
[16] A. P. Gerdelan , M. J. Johnson , C. H. Messom, "Performance Analysis of Virtualised Head Nodes Utilising Cost-Effective Network Attached Storage," http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.77.3585.
[17] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions," *EUROCRYPT*, 2005.
[18] IEEE, "IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices," *IEEE Std 1619-2007*.
[19] M. Liskov, K. Minematsu, "Comments on XTS-AES", http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/XTS_comments-Liskov_Minematsu.pdf, 2008.