

SECURING MACHINE'S LOCAL DATA IN A SHARED ENVIRONMENT

Mohamed Fahmy Tolba mtolba@geganet.com

Ain Shams University, Egypt.

Mohamed Saeed Abdel Wahab wahabms@hotmail.com

Ashraf Saad Hussien ahrafh@acm.org

Mohamed Ahmed Abo El-Fotouh midono1@hotmail.com

Faculty of Computer and Information Sciences, Ain Shams University, Egypt.

Abstract: Securing machine's data in a shared environment became an important issue due to rapid evolution of networks, grid and P2P computing. This paper proposes a solution that can protect machine's specific data (Local data) from disclosure. The proposed solution is based on a novel cipher Tornado. This cipher has a typical 256-bit symmetric key strength but after applying the proposed modification to this cipher; its key strength was raised to 288-bit; this was achieved by adding a second key of length 32-bit. The idea of the proposed solution is to encrypt machine's local data using two keys. The first key is the encryption key and the second one is the machine's print. The result of this operation is that the encrypted data will be intelligible on other machines except the machine that it was originally created on, even if the encryption key is known. This paper proposes a recovery plan from hardware failure or upgrades and an access control application.

Keywords: Dynamic encryption, Tornado, Access control, Block cipher applications, Local encryption.

1. INTRODUCTION

Large-scale distributed computing environments or computational grids, as they are sometimes termed [1] couple computers, storage systems and

other devices to enable advanced applications such as distributed supercomputing, computer-enhanced instruments and distributed data mining [2]. The Grid [3] is rapidly emerging as the dominant paradigm for wide area distributed Computing. The emergence of the Grid applications coincides with that of Peer-to-Peer (P2P) applications such as SETI@HOME[4] or Napster [5].

Nowadays the world of distributed computing has many machines that contribute as a part of the system; these machines could be categorized into two main categories: The first category consists of machines that are dedicated to work on that system (all the data stored on these machines is system dedicated). The second category consists of machines that contribute in more than one system (these machines may hold data that is machine specific). The proposed solution's goal is to secure this data.

The proposed system offers data security to machine's local data (data that is used only by the machine) by using two-key encryption algorithm to encrypt the data. This is achieved by modifying the Tornado [6].

cipher to accept two keys. The first key is the encryption key and the second one is the machine print. This causes the encrypted data to be intelligible on other machines except the machine that it was created on, even if the encryption key is known.

The choice of Tornado over other famous and standard algorithms like Triple DES [7], Rijindeal [8], RC6 [9], Seperent [10], and Twofish [11] has been made due to its dynamic ability and its structure that has a second key generated by its key scheduling. The proposed system combines the serial number of the system hard disk (the one with the operating system installed on) and the serial numbers of all the available processors, to generate the machine print; as these factors are not frequently changed by hardware updates. A recovery plan for hardware failure and upgrades is also proposed.

This paper is organized as follows: First Tornado cipher is presented. Subsequently, the modified Tornado, that accepts two input keys. This is followed by the proposed system. Finally, the recovery plan, the conclusion and the future work.

2. TORNADO

Tornado is a new cipher which has both 256-bit block and key size; it has a novel idea that makes it impressive as it operates in 232 different ways. This enforces blind search over the key space which is computationally infeasible. Tornado consists of 10 round functions called "Storm-functions";

the sequence of applying and choosing the candidate Storm-functions depends on the user's input key that makes Tornado a dynamic key dependent algorithm. In addition to the creative key scheduling algorithm that uses some of the cryptographic core of the algorithm to increase key scheduling security [6], this algorithm has been inspired by the famous Blowfish and Twofish [11] algorithms.

Software implementation of the cipher using C++ language shows a high speed-up factor in comparison to other famous and standard algorithms. Performance study shows that it is faster than the AES [8] by a speed-up factor of 253%. Yet we believe that it is more secure than AES.

The encryption algorithm takes three parameters whose description is found in Table (1). The infrastructure of the cipher is found in figure (2.1), and the pseudo-code is as follows:

```

Function TornadoEncrypt (input:In[],cType[],Key[])
    X=In
    for i=0 to 9
        C=cType[i]-'0'
        StormFunctionC(X,Key)
        Key+=8
    end for
    Output(X)
    
```

Table (1): Tornado Encryption nomenclature

Variable name	Description
In[]In[]	put (256-bit) plain text is divided into 8*32-bit words.
cType[]	A 10 bytes array that represents the decimal digits of iType (e.g. if iType=1324567890, then cType="1324567890"), where iType is a 32-bit integer generated during key scheduling algorithm.
Key[]	Expanded Key array (80*32-bit word) which is used in the Encryption process generated during key scheduling algorithm.

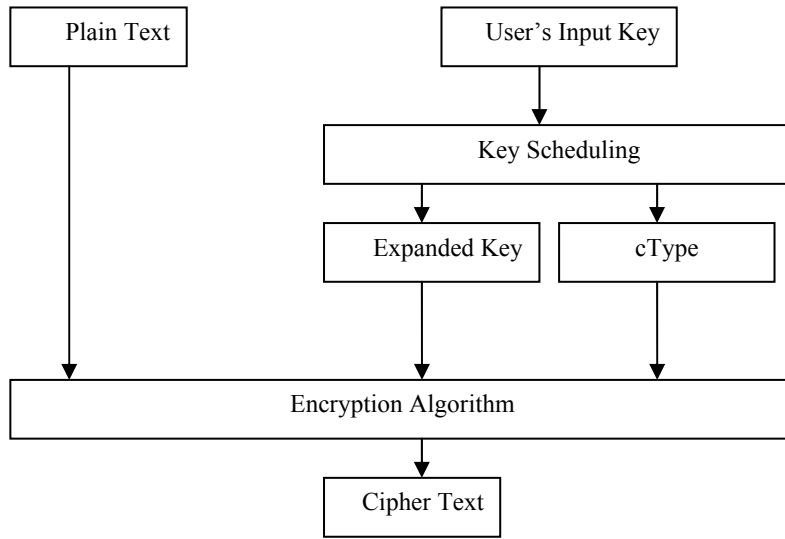


Figure (2.1): Tornado infrastructure

3. MODIFIED TORNAO

The proposed modification to the Tornado cipher is to give the algorithm “cType” byte array as an input instead of generating it from the key scheduling algorithm. This modification will result in a two keyed cipher, where the first key is the user’s input key “password” and the second key is the “cType” array. The infrastructure of the cipher after modification is shown in figure (3.1).

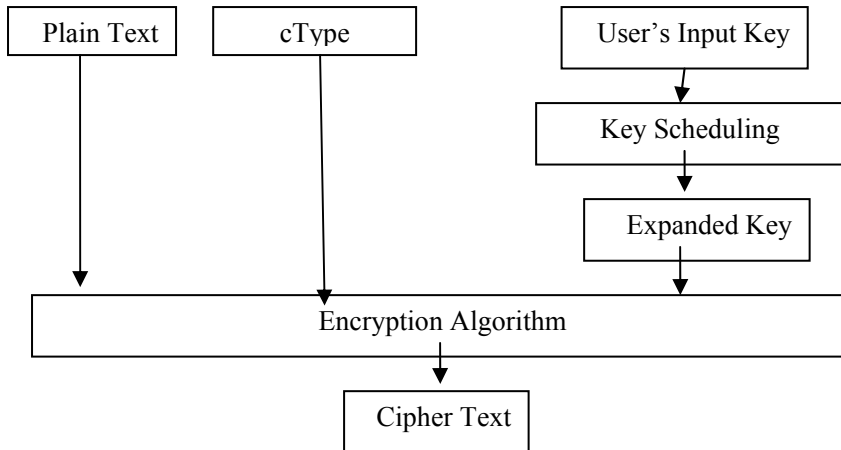


Figure (3.1): Modified Tornado infrastructure

The proposed modification will result in increasing the key strength of the cipher by 232 bit as the “cType” is the decimal representation of 32-bit integer. This increases modified Tornado key length to 288-bit. The “cType” array can be generated from the second key which can be of any length using the same way it was generated in the original algorithm (Note: In the original algorithm the “cType” was generated from the user's input key).

The following pseudo code illustrates the generation of “cType”:

```
Function Generate_cType(input: TheSecondKey[])
    Value=M10(K[0],K[7],K[6]);
    Value=M1(K[1],K[2],Value);
    Value=M6(Value,K[5],K[3]);
    Value=M7(K[4],Value,K[0]);

    iType=Value;
    cType=ConvnetWord32ToBytearray(iType)
```

Note: M”I”s are some of the building blocks of the cipher. Each M-Function accepts three parameters; it uses the last two to modify the first one. These functions are characterized by their high non-linearity. Here are the definitions of the used M_Functions.

Function M1(input : X, Y, K)

```
X=X + Y
X=X - K
X=RotateWord32Left(X, 3)
Output(X)
```

Function M10(input : X, Y, K)

```
X=~X
X=X - Y
X=X ^ K
X=RotateWord32Left(X, 17)
Output(X)
```

Function M6(input : X, Y, K)

```
X=X ^ Y
X=X - K
```

```
X=RotateWord32Left(X, 10)
```

```
Output(X)
```

```
Function M7(input : X, Y, K)
```

```
X=~X
```

```
X=X +Y
```

```
X=X ^ K
```

```
X=RotateWord32Left(X, 11)
```

```
Output(X)
```

4. PROPOSED SYSTEM

The objective of this paper is to develop an algorithm that is capable of protecting unshared data in a shared environment like Networks, P2P, grid computing...etc. The proposed system uses both modified Tornado and machine print to achieve its objective.

4.1. Generating Machine Print

The proposed system combines the serial number of the system hard disk “the one with the operating system installed on the machine” and the serial numbers of all the available processors. Pseudo-code of generating the machine print is presented:

```
Function: GenerateMachinePrints()
```

```
TempString=""
```

```
OsRoot=GetOeratingSystemDrive()
```

```
RootSerial=GetHardDiskSerialNumber(OsRoot)
```

```
TempString=ConvertToString(RootSerial)
```

```
ProcessorCollection=GetAllAvailableProcesors()
```

```
Foreach Processor in ProcessorCollection
```

```
TempString = TempString +
```

```
ConvertToString(Processor.SerialNumber)
```

```
End Foreach
```

```
cType= Generate_cType(TempString)
```

```
Output(cType)
```

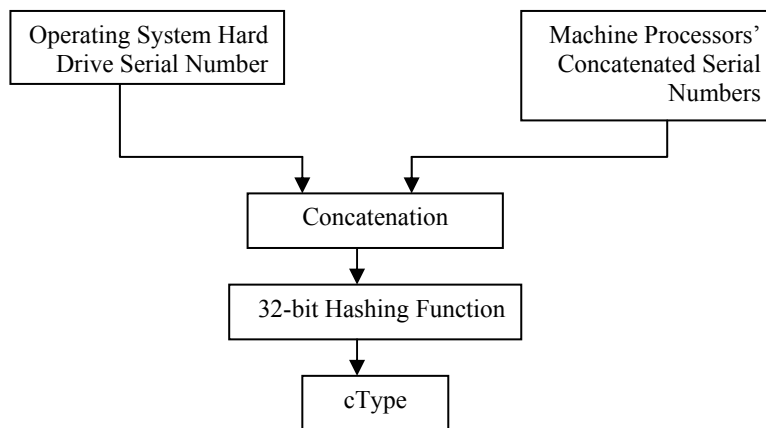
This algorithm first gets the operating system’s drive and saves its serial number in the “TempString” variable, then it obtains a collection of all available processors in the machine and concatenates their serial numbers to the “TempString” variable, this is followed by generating the “cType”

corresponding to the machine print [this “cType” is used as the second key in the modified Tornado algorithm].

The reason for choosing the operating system's hard drive serial number is to eliminate the possibility of the existence of removable hard drives in the system. The reasons for choosing processors' serial numbers as candidates in the machine print are:

- (1) First, each processor has a unique serial number.
- (2) Second, processors are non removable hardware devices so they can represent a permanent candidate in the machine print.

The generation of the machine print is illustrated in figure (4.1), which shows how the cType is generated and how modified Tornado is used for encrypting data.



Figure(4.1) : The generation of the machine print

4.2. Recovery and Hardware Upgrades

Problems that can face the proposed system can be categorized into two main categories. The first category arises due to hardware failure and the second category arises due to hardware upgrades. Fortunately, the solution of the recovery of hardware failure “one or more processors or hard drive containing the current operating system “and that caused by upgrading the existing candidates hardware in machine print are the same. The proposed recovery plan is simple, all what we need to do is store the “cType” generated as machine print encrypted on a removable media. So if any of the previous problems occurred, we can update the encrypted data by modifying its encryption, as shown in figure (4.2).

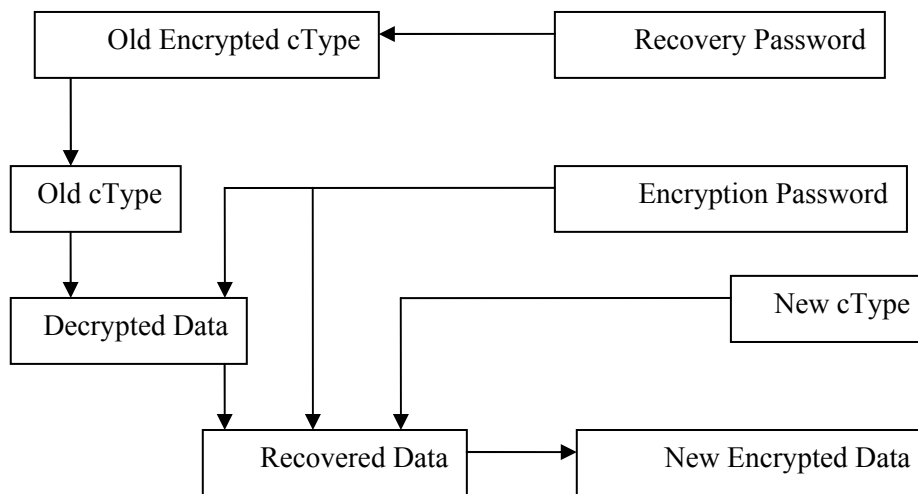


Figure (4.2) : Recovery plan

4.3. Access Control Application

The proposed system can be used to efficiently implement machine dedicated access control application (i.e. access control to a specific machine). The idea is to encrypt a data file “Login file”, which contains the entire user’s account, authority and role information using modified Tornado algorithm after being digitally signed by the machine’s private key (reversed RSA [12] 2048-bit algorithm can be used in this step). Passwords of the user acts as the first key to the algorithm while the machine print acts as the second key. This file is to be stored on a removable media like smart card, flash memory, CD-ROM...etc. As shown in figure (4.3).

To log on the machine, the application will request the “Login file” from the user. The user chooses his login file which is preferred to be with the user on a removable media. Then the application will ask the user for his password¹. The application tries to decrypt the “Login file” using the user’s supplied password and the machine’s print. Then the decrypted data is checked for the validity of the machine’s digital signature. If this verification happened to be true, the application loads from the decrypted file the user’s

¹ It is the user responsibility to use strong passwords to increase the security of the system. Strong passwords should contain at least eight characters from at least three of the following four classes (lower case characters, upper case characters, numerals and non-alphabetic characters).

account, authority and role information; otherwise the user will not be allowed to log on the machine. Figure (4.4) shows the application framework.

Note: the digital signature in the application has two main rules. The first is to be sure that the file was created on the machine that we want to log on. Second to be sure that the data after decryption is valid.

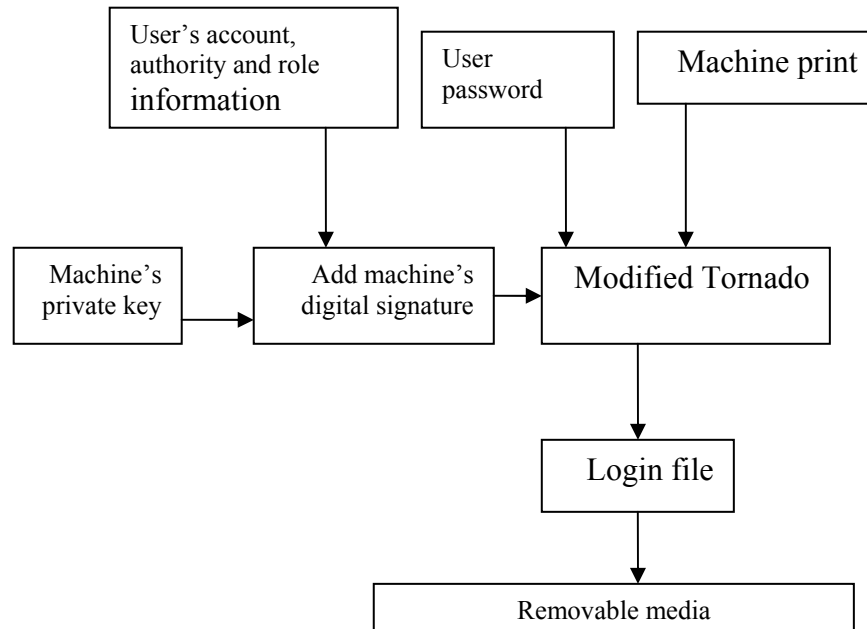


Figure (4.3): Creating the "Login file" process.

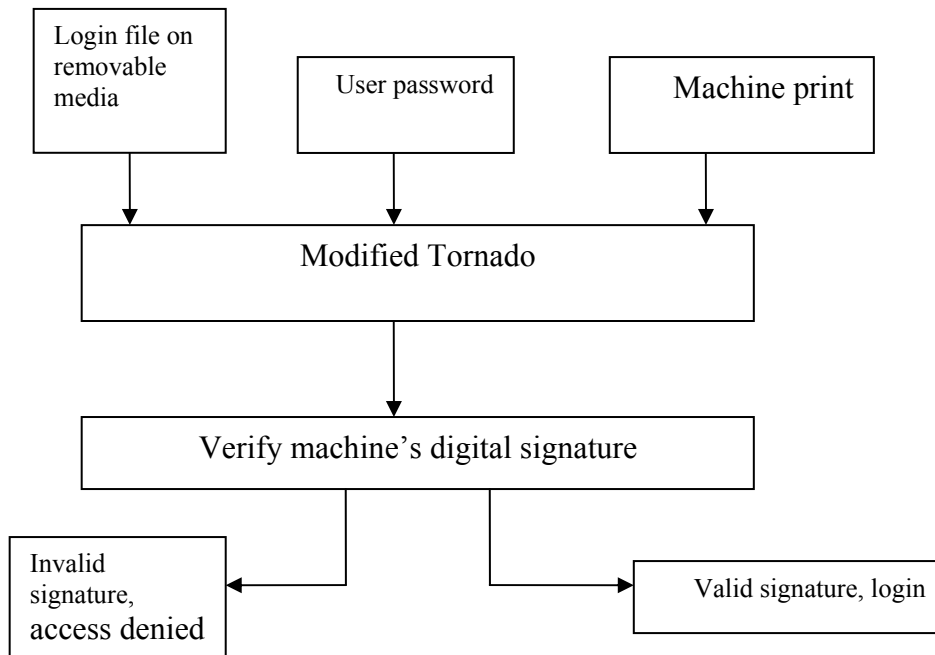


Figure (4.4): Access control application framework

5. CONCLUSIONS

This paper had proposed a system that is capable of securing machine's specific data by encrypting it by local encryption, which means using the machine print as a candidate key in the encryption process. This was achieved by modifying the Tornado cipher to accept two input keys. The first key is the encryption password and the second is the machine print. This caused the encrypted data to be intelligible on other machines except the one that it was created on, even if the encryption key is known. The choice of Tornado had been made due to its dynamic ability and its structure that has a second key generated by its key scheduling. A recovery plan from hardware failure and upgrades was proposed, in addition to an efficient access control application.

6. REFERENCES

- [1] Foster, I. and Kesselman, C. (1998), "Computational Grids: The Future of High Performance Distributed Computing". Morgan Kaufmann.
- [2] Catlett, C. and Smarr, L. (1992), "Metacomputing. Communications of the ACM", PP.44-52.
- [3] Foster, I. and Kesselman, C. (1999), "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman.
- [4] <http://setiathome.ssl.berkeley.edu>
- [5] <http://www.napster.com>
- [6] Tolba, M., Saed, M., Saad, A. and Abo El-Fotouh, M. , "Tornado: A Novel 256-bit Block Cipher".
- [7] Davies, D., Murphy, S. (1995), "Pairs and Triplets of DES S Boxes", in Journal of Cryptology v 8 no , pp 1-25.
- [8] Daemen, J., and Rijmen, V. (1998), "AES Proposal: Rijndael", NIST AES Proposal.
- [9] Contini, S., Rivest, R., Robshaw, M., and Yin, Y. (August 20, 1998) "The Security of the RC6 Block Cipher". Version 1.0.
- [10] Anderson, R., Biham, E. and Knudsen, L. (1998), "Serpent: A Proposal for the Advanced Encryption Standard", NIST AES Proposal.
- [11] Schneier, B. (1998), "Twofish: A 128-bit block cipher", Technical report, Counterpane Systems, Minneapolis, USA.
- [12] RSA Data Security Inc., www.rsa.com.