ISSN:2320-0790

An International Journal of Advanced Computer Technology

A Generic Evolution of Key Using Quantum Cryptography Management for Transcript

Basant Dhakad[#1], *Dharmendra Gwala*[#2], *Manish Sharma*[#3] *and Vivek Shrivastava*[*4]
[#]Department of Information Technology, Student, ITM College, Bhilwara, Rajasthan, India

[1]basantdhakad09@gmail.com [2]dharmudiwan@gmail.com [3]Manish_sharma721@rediffmail.com

[*]Department of Information Technology, Head of Department, ITM College, Bhilwara, Rajasthan, India

[4]Vivek.sri2008@gmail.com

Abstract: Network Security is play very important role in Network system, Because in networked systems, the major security risks occur while conducting business on the Net; The following are some of the security risks occur: unauthorized access, Eavesdropping, Password sniffing, spoofing-spoofing, Denial of Service, virus attack, System modification, Data modification, Repudiation, E-mail bombing. Not With data storage and processing snowballing into a necessity from being an efficient part of any business process or organization, the need for securing storage at various degrees of granularity is gaining considerable interest. The challenge in designing an encrypted file system stems from balancing performance, security perception, ease of usage and enterprise level deploy ability. Often, the most secure solutions may not even be the best solution either due to hit on performance or due to decreased usability. Further, narrowing the trust circle to exclude even hitherto trusted system administrators makes creating an encrypted file system a huge engineering exercise.

Keywords: *Cryptography (*Transcript's*, Quantum), Encryption Keys, Padding Algorithm*

1. INTRODUCTION

Data security has become a very important issue with growing dependence on storage systems and increasing reliance on the internet for communication. The key factors of interest for a deployment of a storage solution are security, performance and usability. Further, administering networks and storage over thousands of systems spread over employees in companies with large head counts is an increasingly difficult task. With growing number of outsourcing industries, such usage conditions are becoming very common. These industries are usually attributed with high attrition rates. A system administrator who may be trusted with proprietary data is a difficult task. Theft of data by employees who may later cross-over to competitors is another concern. While companies commonly impose restrictions like not allowing USB thumb drives in offices, these restrictions have been effective in preventing data thefts and are generally inconvenient.

Security is a greater concern in defense establishments, where secrecy of data is of paramount importance. Most data thefts are found to be rooted at collusion with system administrators. Similar examples are present in other government organizations. Most users in this kind of environment are non-experts who require ease of usability of any new solution provided. The pressing need is to develop storage solutions which may provide a heightened security barrier. All trust the system administrator.

2. RELATED WORK

Approaches to Designing Encrypting File systems

2.1 Encryption Layer

This decides where the actual encryption/decryption operations are performed on the contents of a file during the write/read process from the disk to memory. This would decide where data remains in plain text in the various stages in which it exists on the system, like buffer cache, page cache. The criterion is important because while some of these buffers are per-process,

others are per-system and therefore affects other elements of design.

2.2 Granularity and Encryption

This refers to the smallest unit which uses the same encryption key. This may be whole filesystems as in the case of Cryptographic File System (CFS), or a per-file key as in eCryptfs. It is also necessary to decide what elements are stored encrypted on the disk. The objects in question are file data, metadata, file system based information etc.

2.3 Algorithm

This refers to the choice of using symmetric key or quantum key based algorithms. Symmetric key algorithms provide better speed while quantum key based algorithms provide greater authentication & security by means of robust key management scheme. A trade-off between the two also brings up using a combination of these algorithms.

2.4 Key Management

The design needs to specify the format and location in which the keys are stored. This may be the metadata of the file or the file system information in the uperblock as two examples. Also, an important design decision is whether the keys are per-file, per-user or per-system.

3. KERNEL CRYPTO API

The kernel crypto API additions are essentially designed to serve the requirements of TransCrypt, the implementations are towards developing a generic API. However, omission of certain features from this generic API is either due to the lack of their necessity for TransCrypt or for the sake of a simplistic implementation.

3.1 TransCrypt's Cryptographic Requirements in the Kernel

While several standard user space libraries exist for both symmetric and asymmetric cryptographic operations in the user space the kernel cryptographic functionality has so far been restricted to symmetric key operations. TransCrypt requires support for following cryptographic functionalities.

1. **Symmetric Key Operations.** These operations are already implemented in the kernel. In TransCrypt these operations are for blinding, data block encryption/decryption, securing communication with the PKS, extracting FSK from super block.

2. **Public Key Operations.** TransCrypt requires encryption with public keys for token creation and establishing secure session establishment with the PKS. We also need compatibility with PKCS#1 standards while creating secure cipher text.

3. **Certificate Parsing and Verification.** As public keys are stored in the user space, there authenticity is verified in the kernel by using certificate chains signed by trusted root CAs. We therefore need certificate parsing and verification routines in the kernel.The earlier works had added RSA encryption and decryption services for a fixed modulus size 1024. The implementation took as input a 1024-bit integer A and created a cipher text C by doing the modular exponentiation.

$C = Ae \bmod n$

Here e is the public key while n is the modulus. In our case, n is taken as 1024. However, the input A is usually smaller than 1024 bits.

3.2 PKCS#1 compliance

Asymmetric cryptographic operations require implementation of four routines - encrypt with public key of recipient, decrypt with private key of receiver, sign with private key of signer and verify with public key of signer. While TransCrypt□s current design requires the public encrypt and verify operations, our implementation includes all four functions to provide a generic functionality. We implement two padding schemes as laid out in version 2.1 of PKCS#1.

3.2.1 Encryption using Public Key

We implement two padding schemes as outlined in PKCS#1 version 1.5 and the OAEP padding detailed in version 2.1 of the PKCS#1 document. Once the padding is done we perform a RSAEP to create the cipher text.

3.2.1.1 Version 1.5 Padding

The plain text block (encoded message), EM corresponding to a message M looks like the following:

EM = 0x00 _ 0x02 _ PS _ 0x00 _ M

Here, PS is a padding string of non-zero octets of length k− message length −3

k is the length of modulus in octets. In our case, this is equal to 128(or 1024 bits). The function which performs this padding is called EME PKCS V1 5 ENCODE.

3.2.1.2 OAEP Padding

PKCS#1 version 2.1 lays out a more comprehensive padding scheme called the Optimal Asymmetric Encryption Padding(OAEP). The scheme is shown in figure 5.1. The figure has been borrowed from the version 2.1 of PKCS#1 draft.

The function is called EME OAEP ENCODE. The implementation of the encoding scheme is as follows.

- lHash is generated from label L using SHA-1 hash.

- DB is generated by the following concatenation:

$$DB = lHash \_ PS \_ 0x01 \_ M$$

Here, M is the message to be encrypted. PS is a zeroed padding string of length

$$k - (\text{message length}) - (2 * \text{hash length}) - 2.$$

Again, k is the length of modulus in octets. In our scheme, this translates to $86 - (\text{message length})$

- A random seed is generated of length equal to the length of hash.

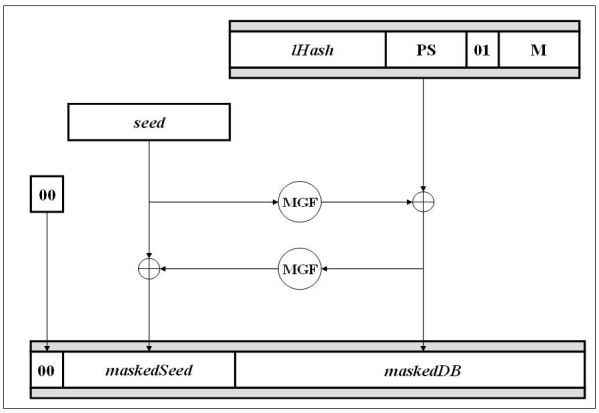

Figure 3.1: Schematic description of OAEP Encoding Scheme. lHash is hash of label L. PS is padding strong of zero octets.

- A mask of length equal to DB is generated using the mask generating function. The mask generating function is outlined later. We call it dbMask. The mask generating function generates mask using seed as the input. We create masked DB by XORing DB with the mask generated above.

$$maskedDB = DB \oplus dbMask$$

- We generate a mask of the length of seed with maskedDB as input called the seedMask. We then create maskedSeed by XORing with seedMask.

$$maskedSeed = seed \oplus seedMask$$

3.2.2 Decryption using Private Key

The message is first decrypted using RSADP operation. The obtained block is then decoded to retrieve the message.

4. APPROACH OF PAPER

In the next chapter we discuss other encrypting filesystems and establish the need for a new encrypting filesystem by using Quantum Cryptography. We outline the architecture of TransCrypt. We also discuss the security framework for TransCrypt. Then in next chapter we discuss about Authentication technique architecture of quantum cryptography. We provide implementation details in later when we develop our thesis. In chapter 5 we take an insight of the future work on TransCrypt System by quantum crypt and finally provide our conclusions.

Current implementation of TransCrypt looks for user's PKS only on the FS and not on WS where user is logged in. This can be solved by registering user's PKS with the FS kernel. FS kernel will store this location and during subsequent file operations, authentication messages will be sent to the user□s registered PKS location.

This section describes an approach to solve the attacks when users access files over network from a workstation using NFS.

4.1 User masquerading attack

NFS is prone to user masquerading attack. Hence processes of the attacker session can have the UID same as that of the genuine user (usr). UID based authentication is not sufficient to protect TransCrypt volume from this attack. FS kernel must have mechanisms to differentiate between the genuine session and a masqueraded session in order to mitigate this attack. To differentiate between these two sessions, we need to establish some unique credentials between the user login process and the FS kernel. During any file access, these credentials will also be sent to the FS. FS kernel will verify these credentials and give access to only those operations coming from a genuine user. Since the masqueraded sessions won't have the correct credentials, it won't get access to the files.

4.2 Man-in-the-middle attack

This attack is solved by establishing a session key between the WS and the FS. All further communication between the hosts will go encrypted with this symmetric key along with message integrity code. The attacker cannot hav e this session key and hence cannot interpret the messages or modify them without being noticed.

4.3 Replay Attack

This attack is solved by sending the file operation responses to the genuine user, encrypted with a freshly generated session key. Only the genuine user will have the correct credential and the session key. These keys (session key and credential) are established at log in time to maintain the freshness so that replay attacks are avoided.

5. QUANTUM CRYPTOGRAPHY

The standard cryptography of today depends on sophisticated algorithms to keep prying eyes from

intercepting and reading our messages. In quantum cryptography, we use the laws of physics for the exchange of keys to protect the confidentiality of our messages over an insecure channel.

There are two requirements for secure data transmission. The first is the secure transmission of the encryption key. One possibility for this transmission is to use quantum particles. The second requirement is an encryption key that consists of random bits. There are two principal quantum encryption protocols, BB84 and entanglement.

5.1 The Theory behind Quantum Cryptography

A photon has a property called polarization and that property can measure. The polarization of a particle is the direction in which the wave is oscillating. Crucially the polarization can be measured either rectilinearly or diagonally.

5.2 BB84 Protocol

BB84 was the first ever quantum encryption schema. Light is composed of discrete packets, called photons. Each photon has an intrinsic property called polarization, which can take one of four values (vertical, horizontal, left and right circular).

5.3 Spin and Polarization

We have already stated that photons exhibit spin. Polarization is the spin propensity of a photon. A photon has an electric and magnetic fields represented by vectors perpendicular both to each other and the direction of travel. The behavior of the electric field vector determines the polarization of a photon.

5.4 Quantum Coding Scheme

As described in the previous section, polarization and measurement of polarization of photons can be done with the use of Polaroid. For the purpose of evolving a simple coding scheme, let us consider only the rectilinear and diagonal polarization schemes. This gives us 4 directions of polarization of a photon. See figure below.

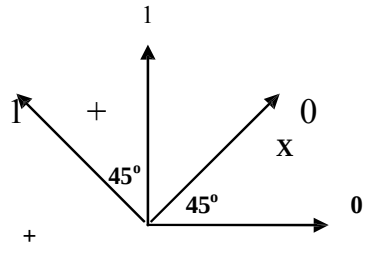

Figure: Mapping of quantum digits to binary digits

Bit 0 = photon with horizontal polarization or by a photon with polarization at 45 degrees to the horizontal direction.

Bit 1 = photon with vertical polarization or by a photon with polarization at 135 degrees to the horizontal direction.

So, using the above qubit representations, a BB84 transmission for the binary 11010011 could look like this:

Table 5.1– Qubit transmission & binary digit selection

Alice : Bits	1	1	0	1	0	0	1	1
Alice : Qubit	↕	↕	↔	↕	⁄	⁄	\	↕
Bob : Scheme	+	X	X	+	+	X	X	+
Bob : Qubit	↕	\	\	↕	↔	⁄	\	↕
Bob : Bits	1	1	1	1	0	0	1	1
Key Selection	√			√		√	√	√

5.5 Quantum Key Distribution: The Basics and Implementation

Quantum key distribution rests on two principles. The first principle is itself one of the fundamental principles in quantum mechanics. The second principle is purely classical in nature.

6. SCOPE OF WORK

The TransCrypt filesystem was designed and a proof of concept was presented in our earlier work. We provide the key management scheme by using QKD (Quantum Key Cryptography) for TransCrypt in this work. This involves design of storage for cryptographic metadata in the filesystem, metadata creation, metadata extraction while opening encrypted files and metadata management. We also provide the kernel cryptographic algorithms necessary for implementing this key management scheme.

We also detail communication with user space utilities which has been done in association with another work of our group. Some of the salient features of this work are:

• Providing support for handling cryptographic metadata.

• Providing a TransCrypt version with full filesystem and file lifecycle.

• Adding associated cryptographic support.

• Providing user space support modules.

• Communication framework with user space utilities.

7. FUTURE WORK

TransCrypt is currently at an advanced implementation stage. Some future goals of the project are enlisted.

- Integrity Support. Adding integrity support for files to ascertain if files have been tampered with or not. The metadata may also be signed to ensure that user tokens have not been changed.
- Data Recovery Agent. For TransCrypt to be enterprise deployable, it should provide recovery agent. The design of the recovery agent has partially been outlined.
- Backup Support. TransCrypt, though implemented over ext3 filesystem, does not handle journaling modes. Investigating and designing backup and recovery is another goal for an enterprise deployable encrypting filesystem.

- Smart Card based PKS. While TransCrypt has from its inception talked about smart card based key acquisition. Smart Card based PKS is a very important future work.
- Group Support. TransCrypt□s design has no provision of Groups, an essential and useful feature of GNU/Linux and has wide ranging usage in enterprises. Group support addition should be done to TransCrypt keeping in view the issues of scalability while still maintaining the features of security, transparency and flexibility.
- Integration with Trusted Platform Module. The recent advent of Trusted Plat- form module (TPM) should be incorporated in TransCrypt. These may be used to avert attacks like malicious kernel image or modules being loaded.

8. CONCLUSION

In this work, we have presented an enhanced architecture of TransCrypt, providing a comprehensive key management scheme. TransCrypt walks over the fine line between usability and security. The design is an attempt to provide flexibility, transparency with a height-ended security paradigm. We identify certain attacks, notably daemon masquerading which Transcript is susceptible to. These attacks are of denial of service nature as opposed to attacks which may lead to data theft.

Attacks leading to data theft are only possible through sophisticated techniques like changing the kernel image, screening the whole memory. We do not consider these attacks a threat for our aim which is to provide an enterprise class encrypting filesystem.

The major contribution of this work is implementation of a full file and filesystem lifecycle. It also presents implementation of user space utilities to support TransCrypt operations. Another contribution is cryptographic additions in the kernel which were needed for Tran-sCrypt. However, these additions have been aimed at providing a common asymmetric API and public key infrastructure in the kernel space.

9. REFERENCE

[1] Mick Bauer. Paranoid penguin: Bestcrypt: cross-platform filesystem encryption. Linux J., 2002(98):9, 2002.

[2] Matt Blaze. A cryptographic file system for UNIX. In ACM Conference on Computer and Communications Security, pages 9–16, 1993.

[3] V Bhanu Chandra. PKI for transcrypt. Technical report, Indian Institute of Technology Kanpur, Kanpur, 2006.

[4] V Bhanu Chandra. Transparent encrypted filesystem. Technical Report, Indian Institute of Technology Kanpur, Kanpur, 2006.

[5] Abhijit Bagri. Key Management for Transcrypt Indian Institute of Technology Kanpur, Kanpur, 2007.

[6] CryptoAPI. The GNU/Linux CryptoAPI, 2003. http://www.kernel.org.

[7] Mohan Dhawan. libacl for transcrypt. http://www.security.iitk.ac.in/home/transcrypt.

[8] dm crypt. A device-mapper crypto target for linux. http://www.saout.de/misc/dm- crypt/.

[9] Satyam Sharma. Transcrypt: Design of a secure and transparent encrypting file system.M. Tech Thesis, Indian Institute of Technology Kanpur, Kanpur, 2006.

[10] Dr. David Knight, Dr. Paul Roach. University of Glamorgan 2004