Encryption 101

Version 2.0 (review in progress)

Encryption 101: Getting Started

To protect data in transit or at rest, encryption is commonly used. In order to accomplish this, proper *key management* is crucial. If a key gets into the wrong hands, unauthorized access to information can result. For data in transit, keys are often generated at the time of use, such as SSL (HTTPS) for web transport, and are temporary. Such data are protected from eavesdropping while in transit, but are no longer encrypted once delivered. Data at rest must be encrypted using a key that has a longer lifespan. It is here that effective key management is crucial. If such a key is lost or destroyed, critical information may become unavailable to authorized personnel. These keys must also remain secret throughout their life in order to be effective.

Encryption is not a panacea. It is complementary to - not a substitute for - other security measures such as authentication, authorization, and access control. Careful analysis of candidate systems is needed to determine encryption's applicability. Many schemes exist for effective encryption, several of which are described below.

Cryptography Basics

Cipher - algorithm

Cryptography - hidden writing, the science of encrypting and decrypting communications to make them unintelligible for all but the intended receiver.

There are two classes of ciphers - block and stream. Ciphers are cryptographic transformations.

Block cipher - symmetric key cipher which operates on fixed-length groups of bits, termed *blocks*, with an unvarying transformation. When encrypting, a block cipher might take a (for example) 128-bit block of plaintext as input, and output a corresponding 128-bit block of cipher text. The exact transformation is controlled using a second input — the secret key. Decryption is similar: the decryption algorithm takes, in this example, a 128-bit block of cipher text together with the secret key, and yields the original 128-bit block of plaintext.

Stream cipher - symmetric cipher where plaintext bits are combined with a pseudo random cipher bit stream (keystream), typically by an exclusive or (xor) operation. In a stream cipher the plaintext digits are encrypted one at a time, and the transformation of successive digits varies during the encryption. These operate in real time and are generally only used once. Stream ciphers are typically implemented in hardware. http://www.lexias.com/2.0/glossary1.html

Types of ciphers:

1. Substitution: used to transform plaintext into ciphertext. Replaces bits, characters or character blocks in plaintext with alternate bits, characters, or character blocks to produce a ciphertext. An example of a substitution cipher is the S-boxes employed in the Data Encryption Standard (DES) algorithm. The S-boxes produce a nonlinear substitution (6 bits in, 4 bits out).
2. Permutation: rearrange bits, characters, or character blocks in plaintext. A simple transposition cipher might be read horizontally, but written vertically to produce the ciphertext. The original letters of the text remain the same, but the order is changed for encryption. DES performs mutations through the use of P-boxes (permutation boxes) to spread the plaintext character over many characters to prevent them from becoming traced back to S-boxed in the substitution cipher.

Most modern encryption systems use both substitution and permutation to achieve encryption.

Cryptosystems - the hardware and/or software implementation that transforms plaintext into ciphertext (or encryption) and back into plaintext (or decryption). A cryptosystem must have the following:

1. the encryption and decryption process is efficient for all possible keys with the cryptosystem's keyspace.
2. is easy to use.
3. the strength of the system is dependent on the secrecy of the keys rather than the secrecy of the algorithm.

Cryptosystems have two elements:

1. Cryptographic algorithm - outlines the procedures used to produce cipher text and plain text.
2. Cryptovariable - the secret value applied to the algorithm. The strength and effectiveness of the system are dependent on the secrecy and strength of this.

Traffic on a network can be encrypted by using end-to-end encryption or link encryption.

1. End-to-end encryption - the information is encrypted once at the original encryption source and decrypted at the destination source. Speed and overall security are advantages. Note only the data is encrypted, not the routing information.
2. Link encryption - requires have separate key pairs for each node that will transfer the message. The information is encrypted and decrypted at every node along the traffic path. Link encryption encrypts not only the message, but the routing information too. Disadvantages include slowing down of message delivery and if a node is compromised, the message can also be compromised.

Symmetric Key Cryptography - also known as symmetric algorithm, secret key, single key and private key, uses one key to encrypt and decrypt data. Advantages include speed, strength and availability. Disadvantages include distribution must be secure, not easily scalable - a different key is required for each communicating pair; and limited functionality due to lack of provision for authentication and non-repudiation. Symmetric key algorithms include DES, Triple DES, Advanced Encryption Standard and International Data Encryption Algorithm and RC5.

- DES - created in the 1970s, DES is a 64 bit block cipher that uses a 56-bit key.
- Triple DES - uses three separate 56 bit encryption keys. The message is encrypted using one key, encrypted again using a second key and further encrypted by using either a first or third key.
- AES - a block cipher designed to replace DES. It is based on the Rijndael Block Cipher.
    - The Rijndael Block - created by Dr. Joan Daemen and Dr. Vincent Rijmen has variable block and key lengths.

- Twofish Algorithm - a symmetric block cipher which operates on 128 bit blocks and employs 16 rounds with key lengths up to 256 bits.

Asymmetric Key Cryptography - also known as asymmetric algorithm, or public key. It uses two separate keys - one to encrypt and a different key to decrypt data. The key pair is known as a public and private key, which are mathematically related, but theoretically the private key can't be derived from the public key. The public key is used to send the message and the private key is used decrypt the message. Its main disadvantage is speed. Advantages include extended functionality and scalability.

- RSA - created by Drs. Ron Rivest, Adi Shamir, and Len Adelman, is a key transport algorithm based on the difficulty of factoring a number that is the product of two large prime numbers.
- Diffie-Hellman Key Exchange - based on discrete logarithms, it is often referred to as a key agreement algorithm. Public keys are shared between sender and recipient. The two keys produce a symmetric key that is only known to the users involved in the exchange. The method is vulnerable to man-in-the-middle attacks.
- Elliptic Curve (EC) - more difficult to compute than conventional discrete algorithms, it is also more efficient because it can use smaller keys and is easily implemented into hardware applications such as wireless devices and smart cards.

Message Authentication - ensures the authenticity and integrity of a message by guaranteeing that it has not been altered during transmission, isn't a repeat of a previous message, was sent from the origin stated, and is sent to the intended recipient. Examples include checksums, CRC-values, and parity checks. More advanced message authentication includes digital signatures and message digests.

- MD5 - a one-way hash algorithm which is a robust popular hashing algorithm. It using a variable size input and produces a fixed-size output (128 bit message digest). Messages are processed in 512-bit blocks.
- SHA-1 (Secure Hash Algorithm) - works similar to MD5. It processes messages in 512 bit blocks and adds padding to a message length to produce a total message length that a multiple of 512.
- HMAC (Hashed Message Authentication Code) (Checksum)) - extends the security of the MD5 and SHA-1 algorithms through the concept of a key digest.

Algorithms and Keys

The two main components of an encryption process are the algorithms and the keys. Algorithms are complex mathematical formulas used for encoding. Keys are random bits that are used by the algorithm to transform the material into its encoded format and back to plain text. Symmetric algorithms use the same key to encrypt and decrypt. Asymmetric algorithms require one key to be used for encryption and a different but related key for decryption.

Encryption Algorithms

Well known encryption algorithms are described in figure 1. With the availability of AES as a strong, standards-based algorithm that is also implemented by many vendors, choosing one of the other algorithms -- RC6, Serpent, Twofish, etc., should go through review and justification. Proprietary encryption algorithms are not recommended because, when weaknesses are discovered in proprietary systems, they are often catastrophic. Algorithms not listed may not be strong enough to provide adequate security.

In instances where hash functions are required to be exclusively unique, MD5 is not recommended. In such cases SHA-256 (or higher) or RIPEMD-320 should be used.

Encryption Strength

Encryption strength is a relative concept. Both the algorithm and the length of the encryption key determine the strength of encryption. Encryption services also perform various cryptographic functions beyond data encryption.

Weak - Algorithms falling into this category are defined by being trivial to decrypt or duplicate without the need of encryption keys or additional information for (example: brute force attacks). Examples of weak encryption are the use of a XOR (EXclusive OR) function, or the UNIX crypt facility. Weak encryption provides only for data obfuscation, and offers no practical level of data security. It only keeps out the curious. Algorithms classified as weak are unacceptable for use in the production University computing environment.

Medium - Algorithms falling under this definition may be susceptible to known attacks or weaknesses, but such attacks will be relatively complex and require non-trivial amounts of computing resources to succeed. Medium strength encryption provides a modicum of data security. These algorithms should only be used as a last resort when algorithms classified as Strong are not viable. In these instances, the encryption keys must be updated every 30 days. Algorithms classified as Medium will be permitted in use at RIT only until 2010.

Strong - Algorithms falling into this category are defined as not having any known weaknesses or susceptibility to any known attacks, and being impossible to break through cryptanalysis methods. Strong encryption refers to algorithms that most effectively ensure the protection and security of data.

Encryption Strength Support Matrix
C = Encrypt/Decrypt
S = Sign (digital signature)
H = Cryptographic Hash
D = Message Digest

Figure 1

Algorithm	Algorithm Type	Algorithm Use	Strength		
			Weak	Medium	Strong
AES (Rijndael)	Symmetric Key - Block Cipher	C			X
RC6	Symmetric Key - Block Cipher	C			X
Twofish	Symmetric Key - Block Cipher	C			X
MARS-(128-1248)	Symmetric Key - Block Cipher	C			X
Serpent	Symmetric Key - Block Cipher	C			X

3DES	Symmetric Key - Block Cipher	C			X
SEAL	Symmetric Key - Stream Cipher	C			X
RC5	Symmetric Key - Block Cipher	C			X*
IDEA	Symmetric Key - Block Cipher	C			X
Blowfish	Symmetric Key - Block Cipher	C	32 bit	256 bit	448 bit
Helix	Symmetric Key - Stream Cipher	C and Authentication	TBD		
Phelix	Symmetric Key - Stream Cipher	C and Authentication	TBD		
CAST	Symmetric Key - Block Cipher	C		64-Bit	128 and above
RC4	Symmetric Key - Stream Cipher	C	40 bit	128-bit	
DESX	Symmetric Key - Block Cipher	C		X	
UNIX Crypt	Enigma	C	X		
ORYX	Symmetric Key - Stream Cipher	C	X		
DES	Symmetric Key - Block Cipher	C	X		
PGP	Public/Private Key	C		1024 bit	2048 bit or greater
RSA	Public/Private Key	C, S			1024 bit or greater
XOR	Bitwise Operation	H	X		
SHA-2 (SHA-224, 256, 384, 512)	Cryptographic Hash	H, D			X
SHA-1	Cryptographic Hash	H, D			X*
MD5	Cryptographic Hash	H, D			X*
RIPEMD - 128, 160, 256, 320	Cryptographic Hash	H, D			X*
RIPEMD	Cryptographic Hash	H, D	X		
Tiger	Cryptographic Hash	H, D			X
Elliptic Curve Digital Signature Algorithm (ANSI X9. 62)	Public/Private Key	S			160 bit
DSA	Public/Private Key	S			Modulus - 1024 bits key size - 160 bits
Elliptic Curve	Public/Private Key	C			TBD
SSL*	Public/Private Key	C, S	40 bit		128 bit and above

*RC5 is considered a strong algorithm (there are no known attacks or vulnerabilities), but there is reason to suspect that it may be vulnerable and its use is not recommended for highly sensitive information or information with an indeterminate lifespan.
*MD5, SHA-1, and RIPEMD - 128 & 160 are considered strong algorithms, but there is reason to suspect that they may be susceptible to frequency collisions (hash duplications) and their use is not recommended in situations where collision resistance is required. In such cases, SHA-2 or RIPEMD-320 is recommended.
*SSL is classified as "weak", "medium", and "strong" depending upon key length. SSL (40-bit) is "weak"; SSL (128-bit and up) is "strong".

Top of page

Key Management

Just like physical security, the strongest lock is useless if the keys are left under the doormat. Security of the key management process for encryption keys is especially important. Together with the review of the encryption method, key management methods must also be reviewed in conjunction with the Information Security Office.

It is important to recognize that many encryption algorithms have the potential to lock a person out of access to their data permanently. So key escrow must be reviewed.

Also, keys may become compromised, and must be revoked. A process for key revocation is essential. A plan and process for movement of all data encrypted with a compromised key to encryption with a new key must be established.

The use of encryption methods for data at rest by individuals, where there is a risk that information would not be available, should be done according to institutional policy, normally only with informed consent. If an approved service exists for key management, it is recommended that individuals utilize that means, or file an exception.

Top of page

Policies

Encryption controls are increasingly mandated by legislation and/or regulations that govern university operations. Sensitive information elements should be identified, with appropriate policy in place to protect those elements. If encryption of data at rest is mandated, data recovery needs to be addressed. Enterprise encryption solutions typically include processes for key escrow and/or data recovery. If a departmental or individual encryption solution is used, management should be made aware that encryption is in use, its purpose, and should possess information on how to recover the encrypted data should the individual who holds the encryption key be unavailable. Links to institutional policies related to encryption are provided below.

- RIT Security Exception Process
- Encryption at the University of California: Overview and Recommendations
- University of Florida Guidelines for IT Workers Regarding Encryption of Stored Data

Top of page

Disk Encryption

Whole Disk definition - Whole disk encryption software encrypts the entire hard drive. The master boot record is altered at authentication boot loader is placed prior to the start of the operating system. The boot loader is not encrypted. Once authenticated to the boot loader the operating system is unaware that the volume is encrypted.

Why Whole Disk?

Whole disk encryption encrypts the entire hard drive. Modern operating systems have a tendency to leave remnants of data throughout secondary storage. Examples of these data remnants are spool files, temporary files and virtual memory/hibernation files. While file/folder encryption can protect confidentiality of data a user is aware of it is unable to protect these remnant data files. Whole disk encryption solutions are ideal for the protection of all data contained on a hard drive.

Whole Disk Encryption Performance - A common method for achieving whole-disk encryption is to implement it as a feature within the device driver stack. Disk device drivers operate between the OS and the physical disk, abstract the disk as a simple array of blocks, and handle all block I/O to the disk. Encryption/decryption happens "below" what the OS sees of the disk, thus is functionally invisible to it and to the applications it hosts. The computations are handled by the device driver during each read/write action, and run on the main system CPU. Some user-friendly whole-disk implementations integrate their key-management systems with the OS' authentication system, permitting a more transparent user experience.

We are beginning to see a move from software toward hardware-based disk encryption, where the cryptography occurs entirely within the disk device itself. As with software driver-based encryption this can be transparent to the OS and applications. The key benefit is that the encryption system is not OS-dependent, and the main system CPU is not burdened with the encryption work. Such devices may deliver performance comparable to an unencrypted disk system. These solutions generally rely on firmware passwords (e.g. "BIOS" passwords), and so it's important to verify if their key management schemes are compatible with your enterprise desktop management system, and with user experience expectations.

Technology process integration - Whole disk encryption goes far beyond simply installing the software on a system. The encryption of an entire hard disk and the accompanying boot loader impact numerous business processes.

- Installation - During the installation of new systems the encryption software needs to be installed. If the old system had whole disk encryption the data will need to be removed from the volume through the operating system or decrypt the hard drive.
- Workstation support - Traditionally support staff could use their credentials to work on systems during support tickets. Depending on the implementation the support staff may not be able to authenticate past the whole disk boot loader. System owners may need to be present to have support staff work on their computers as a result.
- Hardware shop - On campus hardware shops may be presented with challenges similar to workstation support staff. It may be unreasonable to have the system owner present during the entire hardware support session, and escrowed keys may need to be used to access the system.

Criteria for selection matrix - With any software purchase selection there are group of requirements/needs that must be evaluated to make the decision. The following is a listing of some of the common criteria used for whole disk encryption.

- Cost - Whole disk encryption software is typically licensed by user count or by system count. Some systems offer features beyond whole disk encryption that may address other encryption needs and increase the return on investment.
- Performance - Traditionally one of the limiting factors to whole disk encryption, performance is impressive with most solutions. This is still a key factor in selection.
- Centralized management - With hundreds and possibly thousands of nodes receiving whole disk encryption management of the nodes becomes essential. The criteria for a centralized management solution will vary by site but generally need to provide enrollment, licensing, node status, key escrow, policy and authentication. These are basic requirements and many systems go far beyond that.
- Key recovery - With whole disk encryption an organization must control access to keys and be able to recover data in the event the user is unable to or the organization is mandated to for legal reasons. Enterprise systems provide the ability to escrow keys with a centralized server for this purpose.
- Ease of Deployment - Can the solution be reliable deployed remotely, can a user easily follow installation instructions, does the impact of whole disk justify manual installation by a technician
- Multi Platform - Currently available: Windows, Linux and Mac

Reference - http://en.wikipedia.org/wiki/Comparison_of_disk_encryption_software

Other measures needed to make whole disk effective and reliable

- Backups - Loss of data is always a concern when encrypting data. Key escrow helps to mitigate that risk. In the event that a hard drive becomes corrupted traditional means of recovery, such as partial copy or disk imaging will likely fail. The best way to remediate this is to have solid backups of the data on the drive. This may seem like a significant burden to implementation but the loss of one users data due to whole disk encryption can result if the stoppage of the project.

- Screensavers - Whole disk encryption only protects the data from unauthorized access when the system is not authenticated past the boot loader. For instance a user leaving a laptop unattended and logged in will still allow access to data unencrypted. To mitigate this additional risk one must educate users and employ policies to protect access to the operating system. One example of these additional controls is screensavers that require authentication to unlock.
- Hibernation vs. Standby - Whole disk encryption depends on the boot loader to protect the data. If a system is placed into standby and then is brought back online the user will bypass the boot loader and go straight to the operating system. This is due to the fact that the hardware was not truly powered off. In contrast, hibernation writes the contents of ram out to disk and fully powers off the system. At the time the system is powered back on the user is presented with the boot loader and must authenticated before accessing the encrypted hard drive

Authentication models - Whole disk encryption requires the user to authenticate to a boot loader (if the boot drive is encrypted) or an application if the encrypted volume is not the boot drive. This authentication can take several forms.

- Single Sign on - User uses on set of credentials to authenticate to both the boot loader and the operating system. The credentials only have to be entered once and are seamlessly passed to the operating system.
- Unified sign on - Similar to single sign on, the user has one set of credentials for the boot loader and the operating system. The difference is that he user must enter the credentials into the boot loader and operating system individually.
- Separate credentials - The user has a unique set of credentials for the boot loader and the operating system. Each credential must be entered into the appropriate application.
- Multifactor -Multi factor authentication is a compliment to the previously discussed authentication models. Multifactor authentication constitutes two of three factors.
    - Something you know - e.g. password
    - Something you have - e.g. smartcard
    - Something you are - e.g. biometrics, fingerprint

Quality Assurance - whole disk encryption software must meet a high level of quality assurance. As with any widely deployed software solution it is bound to be blamed for numerous orthogonal issues. A solid quality assurance testing plan can help to remediate much of uneasiness posed by the software. Prior to initial deployment a testing plan should be developed and used to identify any impact the software may have on the operating environment. Any time a new version is release the testing plan should be repeated. In addition, identifying a subset of users to first receive new releases will minimize the problems that general end users experience.

Top of page

Public Key Infrastructure

Most colleges and universities make use of encryption on a small scale, encrypting traffic to a small number of Web and e-mail servers using SSL certificates purchased from a certificate authority (CA) such as VeriSign. However, few institutions can afford to purchase large numbers of certificates from a commercial CA. As another example, use of PGP on a small scale is common in many institutions but more than a couple hundred users are difficult to manage and support even with PGP Enterprise. For encrypted e-mail, S/MIME is a more effective long-term solution because it is a standard that is being integrated into more e-mail applications, making it more user-friendly and less costly to support. Rather than dealing with a separate application, users and technical support staff are able to use an e-mail client they are already familiar with.

A few institutions have created their own Public Key Infrastructure (PKI) to support large scale use of encryption. Solutions can be developed entirely in-house or as a mix of commercial and freeware solutions. The decision to build or buy is based upon the needs and goals of the institution and the resources available to provide the infrastructure.

For institutions that have a Windows domain infrastructure, it is relatively inexpensive to create a basic PKI using Microsoft Certificate Server. Using this solution, you can create SSL certificates for Web servers, IPSec certificates for all machines in the domain, and S/MIME certificates for all users. However, these certificates have somewhat limited usefulness because they are only trusted within the institution. Additional planning and effort is required to make internally generated certificates trusted outside of your institution and create a flexible PKI that can persist as the underlying technology changes as described in the following sections.

Creating a Root Certificate Authority Using OpenSSL

The least expensive approach to creating a portable, standard compliant root certificate is using OpenSSL, an open source toolkit available on Windows and UNIX. The following command creates public and private keys for your own top-level certificate authority using parameters set in the configuration file named "openssl.conf". Notably, the certificate authority x.509 extension must be set at this stage (CA=true).C:\> openssl

OpenSSL> openssl req -new -x509 -keyout A:\CA\private\collegeXrootca.key \

> -out A:\CA\ collegeXrootca.cert -config openssl.conf

Once this top level certificate authority has been created, you can create any number of subordinate certificate authorities using Microsoft Certificate Server, certifying keys using the following command.OpenSSL> x509 -in microsoft-subcert.req -out microsoft-subcert.cert \

> -req -days 1825 \

> -CA collegeXrootca.cert -CAkey collegeXrootca.key -CAserial C:\CA\ca.srl \

> -extfile openssl.conf -extensions v3_ca

The "v3_ca" option indicates which section in "openssl.conf" lists the certificate extensions that should be included in this certificate. Again, the certificate authority x.509 extension must be set at this stage (CA=true) or Microsoft Certificate Server will complain that it is not a CA certificate.

Expanding Trust

To expand their trust hierarchy beyond higher education, some institutions outsource portions of their PKI to a globally recognized certificate authority. Starting in 2010, higher education institutions can take part in the InCommon Certificate Services to obtain certificates that are trusted outside the institution.

Top of page

Bibliography

- RIT Security Exception Process
- Encryption at the University of California: Overview and Recommendations
- University of Florida Guidelines for IT Workers Regarding Encryption of Stored Data

#Top of page

Questions or comments? Contact us.