



Req	PCI DSS Requirements	Testing Procedures	Comments
-----	----------------------	--------------------	----------

NetLib Comments: Most of the requirements are procedural. The comments column provides information on how NetLib helps to support the requirement. The answers in this document are in draft form, and may be subject to change.

Requirement 3: Protect stored cardholder data			
3.1-3.3			These are procedural in nature and are not controlled by Encryptionizer.
3.4 Render PAN, at minimum, unreadable anywhere it is stored (including data on portable digital media, backup media, in logs, and data received from or stored by wireless networks) by using any of the following approaches:	3.4a Obtain and examine documentation about the system used to protect store data, including the vendor, type of system/process, and the encryption algorithms (if applicable). Verify that the data is rendered unreadable using one of the following methods:		Encryptionizer supports AES and 3DES up to 256 bit encryption.
<ul style="list-style-type: none"> • Strong one-way hash functions (hashed Indexes) • Truncation • Index Tokens and pads (pads must be securely stored) • Strong cryptography with associated key management processes and procedures. <p>The MINIMUM account information that must be rendered unreadable is the PAN. <i>If for some reason, a company is unable to encrypt cardholder data, refer to Appendix B: "Compensating Controls."</i></p>	<ul style="list-style-type: none"> • One way hashes (hashed indexes) such as SHA-1 • Truncation or masking • Index tokens and PADS, with the PADS being securely stored • Strong cryptography, such as Triple-DES 128-bit or AES 256-bit, with associated key management processes and procedures. 		
	3.4.b Examine several tables from a sample of database servers to verify that the data is rendered unreadable (that is, not stored in plain text)		Data is stored to disk as encrypted and is never decrypted on disk even when in use. Database files can be inspected and determined that the database files contain no unencrypted data.
	3.4.c Examine a sample of removable media (for example, backup tapes) to confirm that cardholder data is rendered unreadable.		Encrypted databases can be moved to other systems and do not become decrypted in the process. Backups directly to tape can also be encrypted. Encrypted databases can only be accessed with Encryptionizer software configured with the same encryption key profile (algorithm, key length and passphrase)
	3.4.d Examine a sample of audit logs to confirm that cardholder data is sanitized or removed from the logs.		procedural - your organization should have a way to sanitize the logs. However, you do have the ability to encrypt the SQL Transaction logs (*.ldf) as well.
3.4.e Verify that cardholder data received from wireless networks is rendered unreadable wherever stored.		procedural - as long as once received the data is stored in the Database protected by Encryptionizer, the data will be encrypted. Encryptionizer does not specifically encrypt data in transmission.	
3.4.1 If disk encryption is used (rather than file- or column-level database encryption.....)			Not applicable for Encryptionizer
3.5 Protect encryption keys used for encryption of cardholder data against both disclosure and misuse	3.5 Verify processes to protect encryption keys used for encryption of cardholder data against disclosure and misuse by performing the following		
3.5.1 Restrict access to keys to the fewest number of custodians necessary	3.5.1 Examine user access lists to verify that access to cryptographic keys is restricted to very few custodians		Procedural
3.5.2 Store keys securely in the fewest possible locations and forms	3.5.1 Examine system configuration files that verify that cryptographic keys are store in encrypted format and that key-encrypting keys are stored separately from data encrypting keys.		Using Encryptionizer - key-encryption keys and data encryption keys are separated from each other. Additionally encryption keys are separated from the data. The Key profile is strongly encrypted (3DES, 256 bit) using a Base Master key that is then semi-randomized. The location of the Base Master key and randomization technique is proprietary. The Base Master key is not stored with the encryption key profile or the database.

Req	PCI DSS Requirements	Testing Procedures	Comments
-----	----------------------	--------------------	----------

NetLib Comments: Most of the requirements are procedural. The comments column provides information on how NetLib helps to support the requirement. The answers in this document are in draft form, and may be subject to change.

	3.6 Fully document and implement all key management processes and procedures for keys used for encryption of cardholder data, including the following:	3.6.a Verify the existence of key management procedures for keys used for encryption of cardholder data	Procedural
		3.6.b For Service Providers only: If the Service Provider shares keys with their customers for transmission of cardholder data, verify that the Service Provider provides documentation to customers that includes guidance on how to securely store and change customer's encryption keys (used to transmit data between customer and service provider)	Procedural
		3.6.c Examine the key management procedures and perform the following:	
	3.6.1 Generation of strong keys	3.6.1 Verify that key management procedures require the generation of strong keys	Strong encryption keys can be generated with 3DES, AES or Blowfish algorithms up to 256-bit encryption
	3.6.2 Secure key distribution	3.6.2 Verify that key management procedures require secure key distribution	The encryption key profile can be strictly associated with a single machine using the "lock key to machine" option
	3.6.3 Secure key storage	3.6.3 Verify that key management procedures require secure key storage	The encryption key profile is stored in a strongly encrypted file using a randomly generated key in a location separate from the database. By default it is stored to the BINN directory of the SQL instance, however it can optionally be stored to another location including removable media or network location. The encryption key profile must be available at the time of SQL start.
	3.6.4 Periodic key changes <ul style="list-style-type: none"> As deemed necessary and recommended by the associated application (for example, re-keying); preferably automatically At least annually 	3.6.4 Verify that key management procedures require periodic key changes. Verify that key change procedures are carried out at least annually	Encryptionizer supports re-encryption of a database from one key to another without having to completely decrypt the data.
	3.6.5 Destruction of old keys.	3.6.5 Verify that key management procedures require the destruction of old keys	Procedural. However, if a new encryption key profile is generated and stored to the same location, the previous key profile is overwritten and destroyed.
	3.6.6 Split knowledge and establishment of dual control of keys (so that it requires two or three people, each knowing only their part of the key, to reconstruct the whole key)	3.6.6 Verify that key management procedures require split knowledge and dual control of keys (so that it requires two or three people, each knowing only their part of the key, to reconstruct the whole key)	NetLib Encryptionizer supports split knowledge of encryption key information using the Permanent Mask Key feature in the Admin Wizard. This allows more than 1 person to enter a portion of the key phrase without exposing another person's entry.
	3.6.7 Prevention of unauthorized substitution of keys	3.6.7 Verify that key management procedures require the prevention of unauthorized substitution of keys	NetLib software has controls (content Hash) that will detect if the key file has been replaced. If the file has been replaced SQL Server will not even start up and no data can be accessed
	3.6.8 Replacement of known or suspected compromised keys	3.6.8 Verify that key management procedures require the replacement of known or suspected compromised keys	Procedural
	3.6.9 Revocation of old or invalid keys	3.6.9 Verify that key management procedures require the revocation of old or invalid keys (mainly for RSA keys)	Procedural
	3.6.10 Requirement for key custodians to sign a form stating that they understand and accept their key-custodian responsibilities	3.6.10 Verify that key management procedures require key custodians to sign a form specifying that they understand and accept their key-custodian responsibilities	Procedural