

Code42 Security

Code42 provides continuous, automatic desktop and laptop backup. Our layered approach to security exceeds industry best practices and fulfills the enterprise need for convenience and flexibility.

SOLUTION HIGHLIGHTS

- ▶ AES 256-bit file encryption
- ▶ AES 256-bit communications encryption
- ▶ LDAP and Shibboleth SSO authentication server support
- ▶ Flexible key escrow policy
- ▶ Auditable, logged restores
- ▶ Automated data retention policies and lifecycles
- ▶ Tamper-proof backup archives
- ▶ Centralized administration and event logging
- ▶ Flexible policy management and enforcement
- ▶ Proxy support

Security context

To understand the security of Code42, it helps to have a high-level familiarity within the context of how it works:

- ▶ Code42 backup begins with the app recognizing file modifications made by the user in real time. Files are analyzed quietly in the background to identify what is unique about the change. Unique information is broken into blocks, then compressed and encrypted symmetrically using a local archive key.
- ▶ The encrypted information flows through a secure communications channel to multiple destinations, as specified by an administrator.
- ▶ Data remains in its encrypted state on disk at the destination. Decryption occurs when an authorized user requests the restoration of data and supplies any required password or archive key.
- ▶ Code42 restore begins when the user or administrator selects files to restore. Encrypted blocks are received by the app, decrypted using the local archive key, decompressed and written out locally to a disk to complete the file recovery process.

Technology

Code42 utilizes pre-compiled Java in a native application. All cryptographic functions rely on the industry standard Java Cryptographic Extensions (JCE) provided therein. There are numerous security benefits to this architecture, including:

Type safe execution

Only secure, type safe byte code is executed, providing protection against buffer overflows and similar mistakes traditionally exploited by attackers.

Open model

Rather than relying exclusively on “security through obscurity,” the cornerstones of our security frameworks are available via open-source for peer review.

Comprehensive security framework

The JCE has been widely reviewed and deployed in countless enterprise applications.

Account security

Code42 can leverage your existing LDAP or Shibboleth server for user authentication. User account states can be updated in real time (pushed) or delayed (pulled), depending on the authentication system used. With this model, Code42 does not store user login information in its database.

Roles

With Code42, a user account may be authorized with the roles below, listed from those with the fewest to those with the most permissions:

Desktop user

Permission to back up to a Code42 server (default).

Code42 user

Permission to access the Code42 admin console. Access is restricted to the users' computers and their data (default).

Org manager

Read-only access to all users in their respective organizations and all child organizations.

Org admin

Read/write access to all users in their respective organizations and all child organizations.

All org manager

Read-only access to all users in all organizations.

All org admin

Read/write access to all users in all organizations.

Admin restore limited

Permission to perform a limited size restore for all devices the admin has permission to view.

Admin restore

Provides permission for user to restore data on behalf of other users within the organizations the admin has permission to view.

Push restore

Permission for the admin to initiate a restore to any device the admin has permission to view.

Server administrator

Permission to edit all system information not reserved exclusively for the SYSADMIN. Read/write access to all users, all organizations, all children and all servers.

SYSADMIN

Read/write access to all users, all organizations, all children and all servers.

In addition, admins with full SYSADMIN permission can create and assign custom roles.

Internal accounts

Accounts can be created by end users without IT intervention, provided the user knows the following information:

Registration Key (aka Organization ID)

A unique and secure 64-bit, globally unique identifier (GUID) containing 16 letters and/or numbers, which is cryptographically secured to prevent a brute force, dictionary-style attack to join the environment.

Code42 server hostname

The resolvable hostname to the IP address of the Code42 server.

Accounts created via this method are limited to Desktop User and Code42 User roles. These roles have no administrative or managerial permissions.

External accounts

Code42 can leverage your existing LDAP or Shibboleth server for user authentication. User account states can be updated in real time (pushed) or delayed (pulled), depending on the authentication system used. With this model, Code42 does not store user login information in its database.

Account lifecycle

Accounts can move from one state to another. Account states include:

Active

Account may interact with its data and computer(s).

Deauthorized

The user is logged out of the deauthorized app installation. Users may reactivate themselves.

Blocked

Account is considered temporarily inactive and blocked from all login activity. Users may not reactivate themselves.

Deactivated

Inactive status. Account is no longer considered in-use and users may not reactivate themselves.

From a security perspective, these states can be transitioned and subsequently trigger pre-defined data retention policies. For example, if an employee's status becomes inactive, Code42 no longer accepts backups from this user, until the user is reactivated. The data is held for 30 days, after which it is deleted.

Data security

Files and their respective metadata are AES 256-bit encrypted using the Archive Key on the source computer. This has several benefits:

Encrypted data is securely transmitted and stored in a proprietary, virtual disk-type structure optimized for security, reliability and performance on the Code42 server, whether on-premises or in the cloud.

Because all metadata is encrypted, there are no hints available as to which data was encrypted and stored.

Data is checksummed after encryption at the source to provide destinations with the ability to detect corruption or tampering without having encryption keys for the original data.

Archive key creation

An archive key must be present on a system before the first backup occurs. Typically, keys are created at the time of installation using either the random key generation feature in Code42 or a custom key provided by the IT department.

Keys are created using a secure, random number generated from Oracle's Java Cryptography Extensions (JCE) framework. This framework is an audited, open-source implementation, proven to exceed industry standard practices.

Archive key storage on app

The archive key is stored on the device in an unreadable state only when the system is authorized for use—such as during a backup or restore session. The archive key is not stored locally when the device is not authorized for these tasks.

Archive key management

There are three methods of managing archive keys in Code42:

AES 256-bit encryption

Uses the account password to encrypt the archive key.

AES 256-bit encryption + password

Uses an additional password to encrypt the archive key.

AES 256-bit encryption with custom AES 256-bit key

A user-provided, AES 256-bit archive key (Code42 app only).

AES 256-bit encryption

In this mode, the account password is used to encrypt the archive key.

Note, the account password is NOT used to encrypt backup data. The account password may be changed at any time.

AES 256-bit encryption archive key escrow

For the AES 256-bit encryption protection mode, the archive key is escrowed on the Code42 server.

AES 256-bit encryption + password

If additional app security is required, the user may elect to symmetrically encrypt

his or her archive key with a user-provided archive key password. This facilitates key escrowing, while preventing administrators from accessing protected data. However, if the archive key password is lost, no one can restore from the backup archive.

Due to the potential for unrecoverable data, Code42 gives the administrator the option to disable this feature.

AES 256-bit encryption with custom key

In this model, the user provides a AES 256-bit archive key. To ensure security in untrusted, hostile storage environments which are not under direct IT control, the key is not escrowed under this mode. This approach requires you to manually manage your own keys, but provides assurance that there is no way a third party can retrieve your data. This is the most secure yet least convenient policy.

Due to the potential for unrecoverable data, Code42 gives the administrator the option to disable this feature.

Communications security

Communications between Code42 app and Code42 server are encrypted with a unique, computer- and session-specific AES 256-bit key. As part of the app deployment process, IT staff will be able to provide the hostname, port and a RSA 2048-bit public key for the Code42 server. We don't rely upon third-party Certificate Authorities and are not vulnerable to attacks that leverage compromised Certificate Authorities. Also, if the Code42 server or app detects corruption, replay or man-in-the-middle attacks, the session is terminated and will need to be restarted

from the beginning. The Code42 app supports network environments that require network traffic to route through a proxy server.

The specific steps and technologies for establishing the connection are as follows:

1. Establish TCP connection

A single TCP connection to a single port, typically port 4287 or 443, is used for all communication.

2. Establish communication

Session key negotiation, as defined by the TLS specification, is used to establish a unique, 256-bit key for both the app and server.

3. App version verification

Code42 apps must meet minimum version requirements to ensure data integrity before validation of identities is allowed by Code42 server.

4. Identity verification

The computer and user identities are verified at the application level before any additional operations are permitted.

5. Establish application session

User and computer have passed all tests, so session is promoted to an application session allowing for backup and restore.

App version verification

Once the communication session has been secured, additional steps are taken to verify the application security layer. Code42 app provides additional information about itself to Code42 server, including:

Version

Which Code42 app version is running?

OS

On which OS and VM environment is the app running?

IP address

What is the IP Address of the Code42 app?

Timezone

In which timezone is the computer located?

The Code42 app must present additional information about itself before an application login attempt will be accepted. The Code42 server logs the information received and verifies integrity. Invalid data results in a disconnect. Valid data results in a “proceed” message to Code42 app.

Computer identity verification

At this point, the Code42 app and Code42 server are still considered “untrusted” and have no rights beyond additional verification. The first step is for the app to present its unique, 64-bit identity to the untrusted Code42 server.

The Code42 server verifies if this identity is:

Valid

Is this a valid identity?

Active

Is this identity allowed to connect?

Unique

Is this identity already connected within the enterprise?

If any of the above is false, the app is immediately disconnected.

If the identity is valid, the Code42 Server acknowledges it by sending its own unique, 64-bit identity to the Code42 app.

User identity verification

Having received the go-ahead, Code42 app now has a secure communication channel over which to communicate. Note, no actual permission to back up (or do anything else) has yet been granted. Code42 app requests account creation/login using the following information:

Registration key

Cryptographically secure, unique identifier for requesting area to back up to

Username

Identity of user

Password (if using native authentication)

The user's password is sent to the authority server then salted and hashed through 100,000 rounds of SHA-512 using Password-Based Key Derivation Function 2 (PBKDF2).

GUID

Globally unique identifier of computer

Deferred request

Request to defer login for this computer (server must OK)

Code42 server receives the registration key, username, GUID, and deferred request information and validates the contents. If all information is correct, permissions are assigned to connection based on user's defined roles. New users only have the right to back up (store) data. Validation is complete and consistent based on Code42 server configuration, including:

Registration key

Is this registration key valid? Known? If not, disconnect.

LDAP

If registration key provided is backed by LDAP services, identity management is deferred to identity management server in real time.

SSO

If registration key provided is backed by Shibboleth SSO services, identity management is deferred to identity management server in real time.

GUID

Is user trying to create an account with a GUID assigned to another user? That would be impossible, disconnect.

User state

Is this user considered valid and "active" in the organization? Or have rights been rescinded?

Computer state

Is this computer still considered "active" and able to back up? If not, take prescribed action (i.e., permanently disconnect Code42 app).

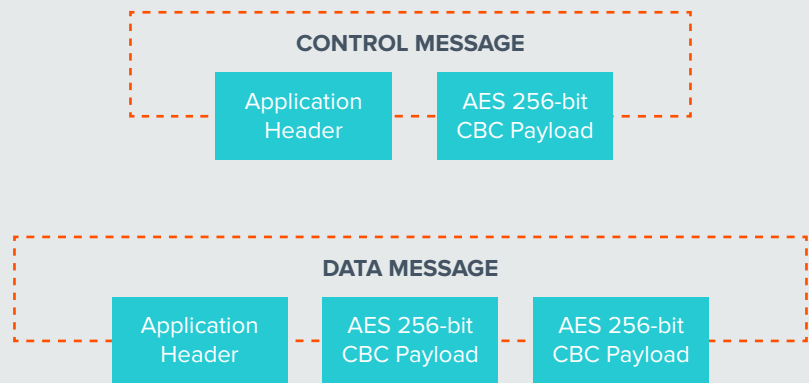
Deferred request

Does this registration key permit invalid login? If not, disconnect.

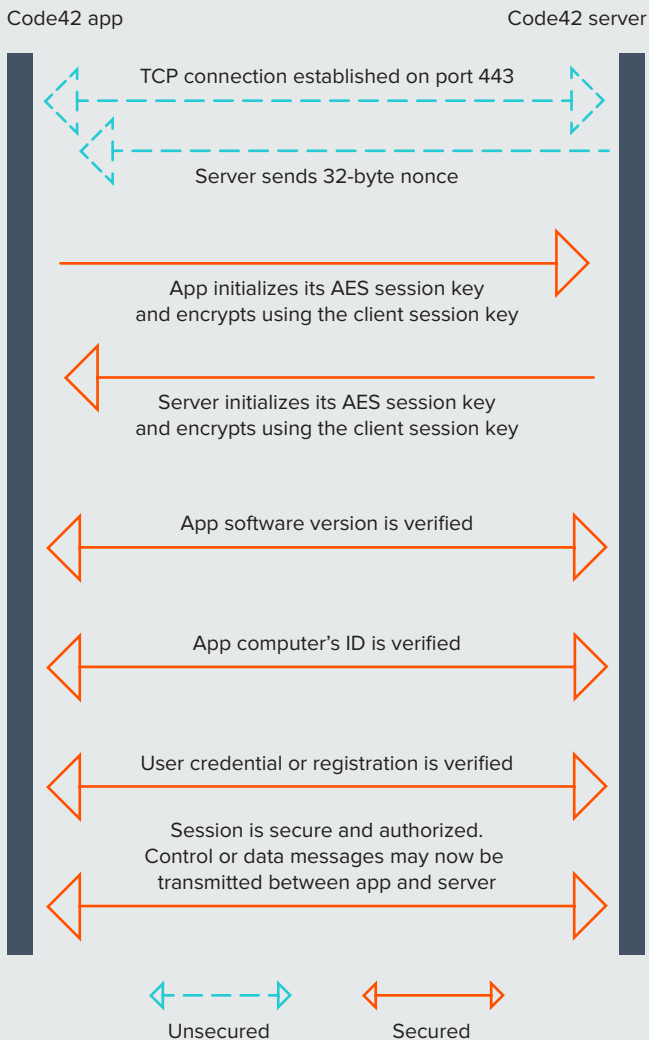
If all criteria are validated, including the optional dependency on a third-party identity management system, permission is granted to the Code42 app to back up and restore. Note, at this time, no additional permissions are granted. The application enforces that the data "sandbox" is unique to a specific user. Ultimately, after all checks have passed, the app merely has the right to store encrypted backup data at the destination.

Control vs. data communications

Control messages, which communicate information, such as verification or status, are constructed of a six-byte application header and an AES 256-bit encrypted payload as specified by the TLS protocol. Data messages, which deliver the AES 256-bit encrypted file data, are constructed of a six-byte application header, AES 256-bit encrypted metadata and the AES 256-bit encrypted file data.



Secure communications sequence



Establish application sessions

Now that app authenticity, user identity and machine identity have been established, the session is allowed access to the backup and configuration layer. The app configuration version is inspected and validated. If the app is “behind” on administrative configuration and/or controls, those controls are pushed down to the app and enforced locally. The configuration includes various runtime operation settings, including:

App UI Settings

All elements of the desktop UI have fine-grain controls permitting the setting and/or locking of various settings.

Destinations

Both permitted and required destinations. This provides administrative controls over what is backed up and to where.

The application session layer remains persistent and connected for the duration of the session. The advantage of this approach is that additional configuration and/or statistics are able to be moved in real time, independent of the backup layer.

Web security

Code42 provides a web administration interface to manage backup for all Code42 apps. Users may also be allowed to restore their files via the web interface. The web interface can be configured to require HTTPS for all communication.

HTTP session creation

Session IDs are generated using a secure random number class and hashed with a secure hashing algorithm.

HTTP session communication

Session IDs are sent in the header of the HTTPS GET request.

HTTP session use

Sessions identify the end user for duration of session to avoid having to repeatedly authenticate for each request. Each request is validated against the current active session.

HTTP session expiration

Sessions expire after 30 minutes of no activity.

HTTP input validation

All data input into Code42 forms is validated and scrubbed of any HTML tags (including SCRIPT) to prevent XSS, XSRF attacks. Additionally, any data passing through to database is merged with previously compiled SQL statements to prevent SQL injection attacks.

Database security

Code42 contains an embedded database to facilitate configuration storage, usage reporting and identity management. The database is not accessible via a network socket and is only available to the Code42 application. Encrypted copies of the database are dumped automatically to available storage points. Encrypted data keys are only present in data stream if key escrowing policy is in effect (refer to encryption key security).



FOR MORE INFORMATION: [CODE42.COM/CONTACT](https://code42.com/contact)

CORPORATE HEADQUARTERS | 100 WASHINGTON AVENUE SOUTH | MINNEAPOLIS, MN 55401 | 612.333.4242 | [CODE42.COM](https://code42.com)