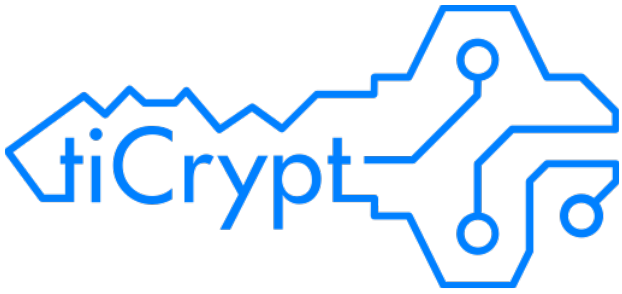


tiCrypt Features

Wednesday, February 3, 2016

Author: Thomas Samant, CMO
info@terainsights.com

tera insights

Overview

This document provides an overview of tiCrypt's major features. These features have been researched and designed via our partnership with the University of Florida. Together with UF researchers and UF security and compliance officers, we created pertinent features for research groups and compliance officers at universities. For more information or a demo please contact thomas@terainsights.com.

Secure File Sharing

File sharing in tiCrypt is made simple and secure through the utilization of public/private key encryption.

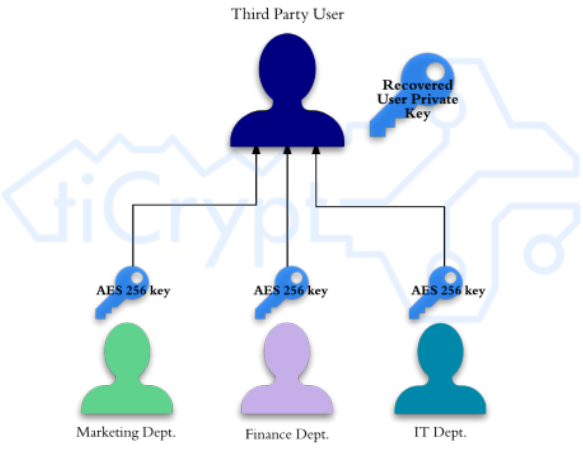
The Process

- Each file, upon creation, is encrypted with a unique, independently generated AES-256 key.
- These keys are then encrypted with the public keys of the users (and groups) that have access to the file and are then stored on the server fully encrypted, unable to be accessed unless decrypted and downloaded by a user with a permissible private key.
- Those that have been given the permissions to share the file (or add others to groups) are then able to add the public keys of users they wish to share the file with, giving those users the ability to decrypt and download the file from the server with their private keys.

- Even if attackers are able to gain access to the servers where the files are stored, the files will remain worthless given the attackers will not have the proper private keys to decrypt the files before downloading.

Key Escrow

Having a key recovery system that is too easy poses an enormous security risk to any system. With this in mind, tiCrypt uses an escrow mechanism designed to require the cooperation of multiple trusted users to protect against key loss, and ultimately, data loss. Specifically, tiCrypt requires a minimum of three escrow keys to recover a user key, but this number



can be customized. More information can be found in the *Escrow Manual*.

How does it work?

Each escrow key needs to be signed with the site key (versus being signed with a normal admin key) and can be used exclusively for escrow activities.

A user's key is placed in escrow using the following mechanism:

- When a user's key is escrowed, the escrow keys with their digital signatures are downloaded from the server.
- These digital signatures are verified against the site key (which is verified by the hard-wired key).
- An AES-256 random key is generated for each escrow key, and then encrypted with the corresponding escrow public key. These encrypted keys are then stored on the server.
- The combination of the AES-256 keys, once for each escrow key, is used to encrypt a user's private key, and then stored on the server.

Successfully Recovering a Key

To recover a user's private key successfully, all escrow key holders transfer their part of the key to a designated third-party. A key CANNOT be recovered if any of the escrow keys are missing. Moreover, only the designated third-party can recover keys, the escrow key holders may not.

Detailed Monitoring

tiCrypt gathers performance and usage characteristics that can be viewed in the monitoring dashboard. Administrators can effortlessly obtain a global, accurate view of the system.

Real Time Monitoring

Administrators can personalize the monitoring screen by selecting various measures as a gauge, a graph, or in table format. These can be further customized based on Admin preferences.

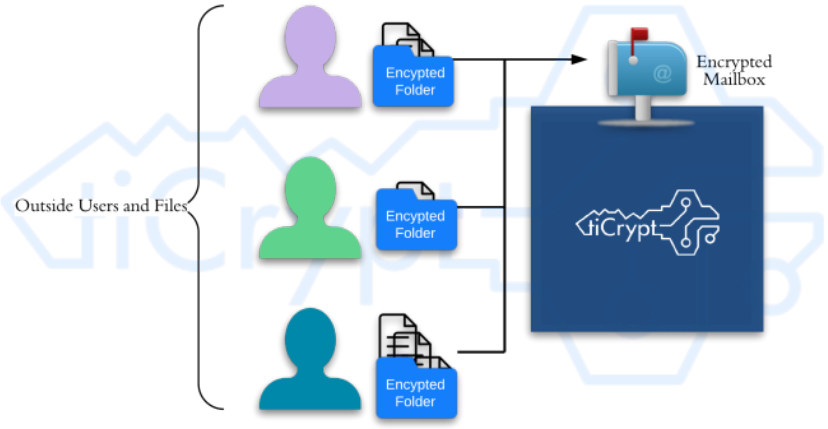
Monitoring of Virtual Machines

Administrators also have an overview of the virtual drives and virtual machines that are running on the user's' behalves.

External Collaboration

There are many situations in which enterprises collaborate with those outside their internal system, leaving many opportunities for files to be exchanged insecurely during communication.

tiCrypt offers a simple collaboration tool that enables tiCrypt users to send the address of an encrypted tiCrypt folder and their public key to individuals outside the system, using the *mailboxes* service. Files can then be encrypted on the outside collaborators' computers using the public key, and the encrypted files can finally be uploaded worry-free to the mailbox, where only the user with the appropriate private key can decrypt them.



Secure Virtual Machines

A key component of tiCrypt is the ability to launch virtual machines on behalf of users within the secure environment. Virtual machines come pre-installed with a number of security features, and will usually be set up on a private network only visible to the tiCrypt server, which will proxy any user interaction with the VMs. While this proxied interaction was previously done through SSH, the new *FIPS 140-2* standard has forced us to innovate. All VM connections are done through in-house forwarding written directly on top of *OpenSSL*'s TLS functionality, which is fully FIPS 140-2 compliant and actually allows for more access control than SSH did, meaning users have access to explicitly specified VM features and nothing more. All user interaction with VMs is monitored and logged by tiCrypt just like other areas of the system. Currently, we have Linux (CoreOS) and Windows images with varying available memory and number of CPU cores. Administrators have the ability to manage VM use, but have no access to the VM itself.

Secure Virtual Machines in tiCrypt

To work with virtual machines (VMs), users download a pre-built application, *tiLauncher*, which may be fed both *deployment* and

application files. Deployments are essentially signed configurations specifying how to interact with tiCrypt and certain VM's. Application files are signed ZIPs containing tiCrypt web interfaces, and are automatically kept up-to-date using trusted repositories specified in respective deployments.

To connect to a VM, the user has to be logged in to the main server, and request to launch a VM (or reconnect to it). When initializing the VM, the server injects the user's public key, while the VM informs the main server of its own public key, which is returned to the user. Also, a single port is opened on the firewall, such that the user (but nobody else) can open a connection with the VM. Using the public keys to encrypt communication, the user and VM negotiate another key to encrypt communication.

Virtual machine images contain the communication protocols to communicate with the Java app. Linux images run CoreOS and are generated with Docker, while Windows images are built by hand. Just as the Java app, the VM images are digitally signed so that the designer can be held accountable for the code in the image.

Multi-User VMs

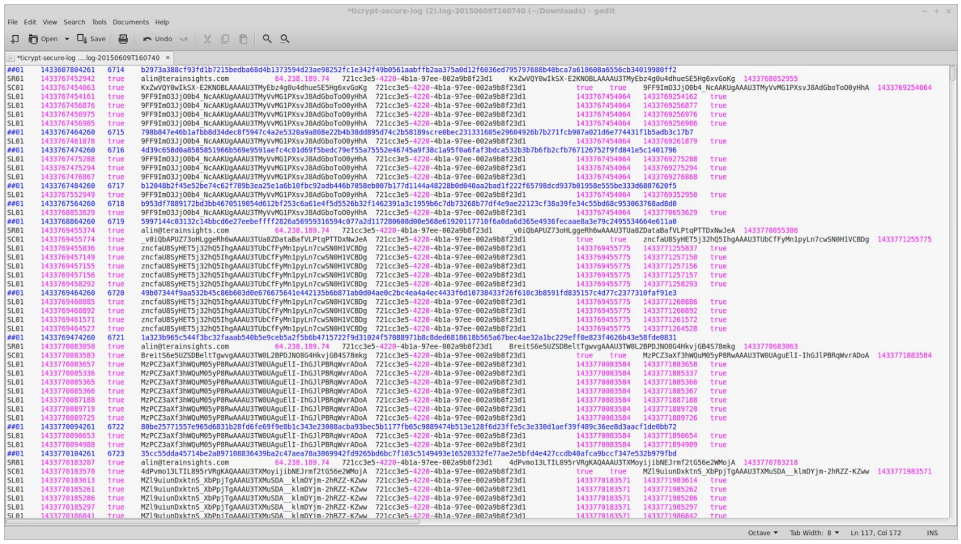
Because all user interaction with VMs is passed through tiCrypt to a proprietary controller running on the VM, the VM is able to completely control which tiCrypt users have access to certain areas of its filesystem, what ports they are allowed to run applications on, etc. While VMs are started up by individual users, they can share access to their VM with other users through tiCrypt, and specify privileges. VM owners can, for instance, declare a particular folder on the VM's home drive to be available to user A and B, but not C. These interactions mean a full ecosystem can be run within a single VM, with the owner in control at all times, of course.

Full-scale Auditing

While it is important to protect files against rogue employees or outside attacks, it is equally important to provide an audit trail of who accessed what information. Basic audits are mandated by most regulatory agencies. tiCrypt goes beyond this minimum requirement and offers detailed, secure audits together with the forensic tools to analyze the logs.

Unforgeable Logs in tiCrypt

Using secure hashing (SHA-256), the log entries are chained much like Bitcoin transactions. Knowledge of the hashes ensures that the history cannot be changed by admins or hackers. The picture at the left shows a snippet of a tiCrypt log. Notice how the hash entries (with prefix ##01) are intertwined with the normal log entries. Should anybody change the log entries, the hashing of the modified block will have a different hash thus the change is immediately detectable.



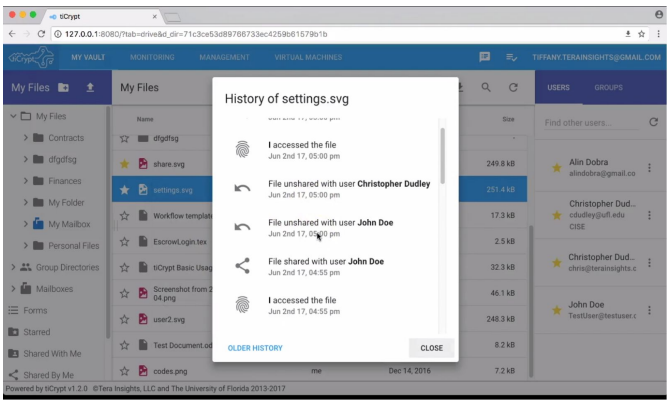
Logs can be analyzed and a complete history of the system can be re-created with relative ease. Our log analyzer allows for creation of easy creation and execution of queries.

Events Audited

50+ events are monitored in the Audit Logs, giving the System Admin a full profile of events.

Examples:

- Session actions: request, challenge, lookup, delete, certificate session, sub session
- Directory actions: create, delete, rename, add/remove entry
- Virtual machine actions: create, delete, register, proxy
- Virtual machine drive actions: attach, detach, create, delete, add/get/delete key
- File actions: create, delete, add/get/delete key, read/write chunk
- Group actions: create, delete, add/remove member, modify group/member
- Storage actions: read, write
- User management actions: add, delete, modified



Data Confinement

Compartmental Design Through Encryption

With tiCrypt, each file or user drive gets encrypted with a unique AES-256 key. These keys get encrypted with the public keys of the users (and groups) that have access to the file. With end-to-end encryption, the decryption is performed before moving the data onto the server (and the decryption takes place at the destination), all data and keys on the server are encrypted. Since a file has to be shared to be available to another user, access to the data can be carefully controlled by the data owner.