PatientKeeper®

TECHNICAL BRIEF

PatientKeeper Platform Extends Secure and Auditable Encryption Standards to Mobile Devices

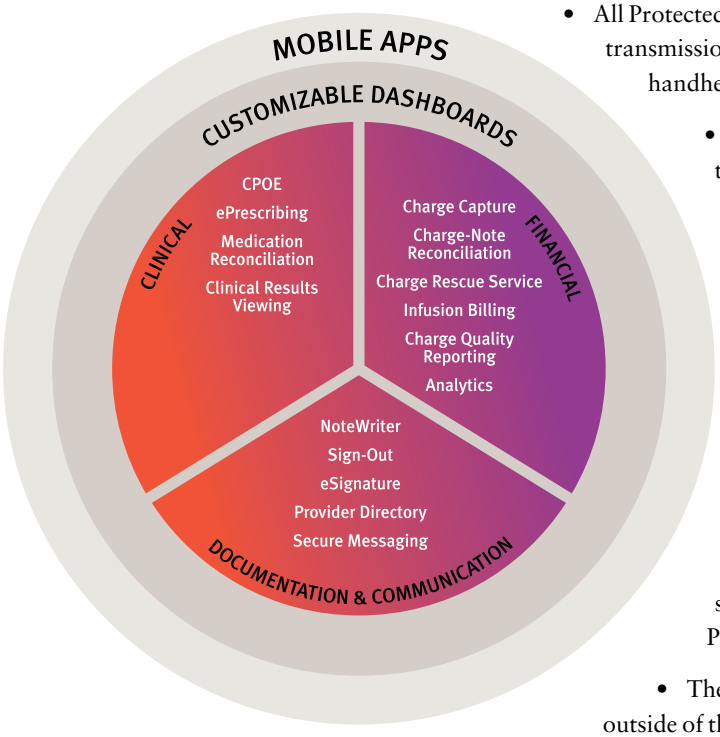
PatientKeeper's Mobile Platform

The *PatientKeeper Platform*™ is a sophisticated framework that enables healthcare providers to aggregate data from multiple systems into a single, native application on Apple iOS and Android phones and tablets. Physicians work from within this integrated, mobile application in a fashion that is meaningful and efficient, sharing patient context throughout their workflow.

PatientKeeper also embraces, extends, and enhances existing information systems infrastructure by allowing clinicians to access and update information using mobile devices. Moreover, the PatientKeeper Platform helps to preserve an institution's IT investments.

HIPAA Security and Patient Confidentiality Support

The PatientKeeper Platform has been designed to fully conform to HIPAA guidelines for ensuring patient confidentiality and privacy. Key features of the application include:



- All Protected Health Information (PHI) is encrypted on the server before transmission to the handheld and remains encrypted until the user on the handheld views it from within the application.
- PatientKeeper uses Advanced Encryption Standard (AES) technology to ensure the highest possible level of security without sacrificing system performance.
- Users must enter a PIN/Password in order to access the application or can utilize the biometric (e.g. fingerprint recognition) security on the device. If a user enters an incorrect password more than a client-specified number of times, the system can lock the user out of the application.
- The system can be configured to "time out" after a client-specified period of inactivity. After timing out, the user must reenter the PIN/Password to gain access to the system.
- PatientKeeper encrypts the entire database stored on the synchronized device. Decryption only occurs from within the PatientKeeper mobile app.
- The application prevents users from copying data to another application outside of the framework. This prevents any PHI from leaving the PatientKeeper security environment.
- PatientKeeper maintains a complete set of logs which record access to PHI. The access is tracked down to the individual result level (e.g. a specific lab test on a specific patient).
- Security and configuration is maintained centrally and pushed to each device so that a security administrator can set access levels, device timeouts, etc. as well as lock a user out of the system without having to physically touch the device.
- PatientKeeper supports remote wiping of synchronized data.

PatientKeeper Security Framework

The PatientKeeper Security Framework is built upon modern, state-of-the-art encryption standards, combining AES and TLS/SSL to provide high-grade security along the entire data pathway. This data can be connected with a backend server and remains secure and auditable along the entire data synchronization path, from server to handheld device.

There are three elements to any security system: authentication, authorization, and data encryption. Authentication and authorization provide security, and data encryption provides secrecy.

Authentication is a mechanism for proving your identity. Authorization grants permission based on that identity, and data encryption allows you to do the entire process without being overheard.

For example, when writing a check at the grocery store, authentication is the process of showing your drivers license. Authorization occurs when the clerk contacts your bank to ensure that you have the funds. Encryption would be the equivalent of walking into a locked room to perform the entire transaction.

Advanced Encryption Standard (AES) and TLS/SSL

PatientKeeper utilizes the cryptographically-secure hashing algorithm SHA-256 for generating symmetric keys. User keys are derived from the user's password (P) and user name (U). Using the user name as salt makes dictionary attacks much harder and more computationally intensive than a brute-force attack. User keys are never used to encrypt data. They are only used to encrypt the key store (KS). Symmetric keys are extremely fast for encrypting and decrypting data, but both the sender and receiver must share the key, reducing security. Public/private keys are extremely secure for encrypting symmetric keys, but are typically slower to use when encrypting/decrypting data. Therefore, the AES algorithm is used for data transmission, and a public/private key infrastructure is used for encrypting all keys and passwords in the PatientKeeper Security Framework.

The use of a public/private key infrastructure for highly secure management of symmetric keys with very fast performance for encrypting and decrypting data is known as a *hybrid* cryptography system. Using hybrid cryptography in combination with a very fast public/private key encryption algorithm (2048 RSA) is the method of choice for mobile/wireless devices.

PatientKeeper Uses Layered Encryption Standards to Provide Triple Encryption for HIPAA-Sensitive Data on a Wireless Network

To provide an extra layer of security, the PatientKeeper Security Framework uses TLS/SSL for data transport on a wireless network. This is the standard security mechanism for support of financial and other confidential transactions on the Internet.

Frequently, the PatientKeeper Security Framework secures the wireless infrastructure by running TLS/SSL on a wireless network on top of the WPA2 standard used by 802.11 wireless routers. HIPAA-sensitive data on a wireless network is encrypted first with WPA2, second with SSL, and third with AES, resulting in the highest level of secure transmission of healthcare data in the industry.

Moreover, additional features support the unique data sharing needs of mobile healthcare professionals. PatientKeeper has enhanced the clinical user's experience on mobile devices, by allowing the user's unique clinical data repository (CDR) password to be used to access backend clinical data in a secure fashion.

The PatientKeeper Platform layered security infrastructure for mobile wireless devices supports encryption of the data on the mobile device, encrypted transport layered on top of heterogeneous wireless security, server authentication of the individual and the device, and determining user level of authorization of access to data on one or more backend CDR or financial systems. In addition, there are audit trails on the handheld device, audit trails on middle tier PatientKeeper Application Servers, and audit trails on the backend CDR and financial systems. Any security breach can be immediately traced to a specific device and user.

Initial Key Setup and Device Provisioning on the PatientKeeper Platform – User Perspective

The institution provides the user with a device and/or the user owns a device that contains no patient data. It is configured for Application Server access and has the program on it that supports synchronization. When the user logs into a device for the first time, the device needs to connect to the network so that the application server can provision users and populate the device with data. The user completes the following steps to prepare their device for encrypted data:

1. If the client does not have the server public key, the client requests it from the server.
2. To complete the requirements for a two factor authentication, the server then sends its public key along with the device ID configuration setting. The server can be configured to require the device ID (D_{id}) to be comprised of the storage card serial number, identifying elements of the device itself, or both (something you have). The device ID is cryptographically secure hash (SHA-256) of a server-specified combination of identifying elements. The server may specify that the ID include information from the memory card, the device, or both. The information used from the memory card is the volume serial number. The information used from the device is a text description of the device. The text description will include the device name (if available), the platform, and the ROM serial number (if available), and the 'user name' of the device (if available). The device ID is used to make sure that the key store has been assigned to the device on which the client software is running. This provides better security by requiring that the user know the user name and password (something you know) as well as requiring the user to access the data on a particular device. A key store cannot be copied from one device to another and be accessible on the new device.
3. The client then sends the user name and password, the unique device ID, and a text description of the device to the server. The text description will include the device name (if available), the platform, the ROM serial number (if available), and the 'user name' of the device (if available) and the serial number of the storage card. All of this information will be encrypted with the server's public key.
4. The server then verifies that this login information is correct.

    a) If it is not correct, the server sends a failure message.

    b) If the login information is correct:

    - The server calculates the device key, assembles the information for the key store and encrypts it with the user key.
    - The server escrows the device key (D_{sym}) and the user key (U_{sym}).
    - The server sends the key store (KS) to the device.
    - The server checks to see if the user already has another device registered. The server may be configured to allow a new device or to reject a new device until the currently registered device is unregistered.

Device Key Revocation

A user key is considered revoked when that user's login credentials change. A new user key is generated when that user logs into the server with the correct credentials. A device key is revoked when any user of a device has had their credentials revoked either because their back-end password has changed or their account has been deleted. Since the device operates while disconnected from the network, there is no way to validate that a key store has been issued by the server at device login time. Revoking a device key prevents a malicious user from storing a key store containing a revoked user key off of the device and then copying it back to the device after synchronization and logging in with the old credentials. When a device key has been revoked, the sequence of events is as follows:

1. The device connects to the server and sends its device ID encrypted with the server's public key.
2. The server determines that the current device key (D_{sym}) has been revoked.
3. The server calculates the new device key D_{sym}, and generates the key store (KS) based on the authorized user of the device.
4. The server sends the key store to the device.
5. The server sends the pre-encrypted data to the device $D_{sym}(DB)$.
6. The server closes the connection.

User Login

The following describes the typical login process from an end user's perspective:

1. The application starts and displays the login screen.

   a) The application checks to see if there are any key stores available. If none exists, then the device must be provisioned (see Device Provisioning).

   b) The user name (U) and password (P) are supplied to the hand held application through the login screen. The application checks to see if the key store (KS) for the user exists by hashing the user name and looking for a match.

   - If it does not match, the login fails. The application goes back to the beginning of the login procedure.
   - If it does match, the application generates a key (U_{sym}) from the user credentials and then decrypts the key store $U_{sym}(KS)$.
   - The application then verifies that the key store has been decrypted successfully by checking that it contains 'user=[U]'.
   - If the key store did not decrypt successfully, then the login has failed and the application goes back to the beginning of the login procedure.

   c) If there has already been the server configurable number of failed login attempts, the application looks for a login failure preference setting. This setting is encrypted with the server's private key. The application decrypts this preference.

   - If the preference indicates a complete reset, the application deletes the database and the key store on the device. A full synchronization will be required to use the device again.

2. The application then creates a static variable to store the device key in memory. The device key itself is not passed from function to function but is accessible to routines that need it as close to the encryption/decryption of data as possible.

Normal Device Synchronization

Normal device synchronization takes place after the device has been synchronized once before by an authorized user. In this state, a device may synchronize with the server without requiring an authorized user to be logged into the device. All sensitive data is pre-encrypted with a symmetric key that belongs to the device and is protected by the user's symmetric key on the device. The following describes a typical synchronization:

1. The device contacts the server with its device ID.
2. If the server recognizes the device ID, it sends the OK to continue.
3. The device sends either the entire database or new data in database file format.
4. The server processes the data, generates the data and sends it to the device along with an updated key store collection (KS).
5. The device updates the database and replaces the key store collection with the one received from the server.

Initial Key Setup on the PatientKeeper Platform – Technical Perspective

The PatientKeeper security package provides authentication, message verification, and data encryption using a combination of public and symmetric key cryptography. Specifically, 256-bit AES in CBC mode is used for symmetric encryption of data on the device and 2048 RSA for public key encryption. In order for the Application Server to run, the server must have a public/private key pair.

The PatientKeeper Platform was designed to make it simple and easy for IT staff and the user to set up these keys on a secure device and on the Application Server. On initial loading of software and data on the mobile device, the end user's Personal Identification Number (PIN) is entered on the mobile device to generate a secure user private/public key combination and a symmetric key for fast data encryption/decryption. The PIN is used to decrypt the private key. The PIN is never stored on the device. The symmetric key can only be decrypted by the user private key.

Device authentication is established through Application Server receipt of a message signed by the user private key. The user private key only exists encrypted on the user device. The user PIN must be entered on the user device to decrypt the user private key to sign a message sent to the server, guaranteeing that the message came from a unique device initiated by entry of the user PIN on that unique mobile device.

User authorization to access or update data on the backend CDR is managed by the backend CDR security infrastructure and not by the Application Server. The encrypted user backend CDR password is signed by the user private key and sent to the Application Server. By verifying the digital signature, the server guarantees that this password came from a particular user's device. The user backend CDR password is decrypted and then transmitted to the backend CDR. The backend CDR then completes or denies the requested authorization for data access. The Application Server never stores the backend CDR password.

The mobile device has the following encrypted keys:

- User symmetric key to encrypt the device key (with username and PIN)
- Device symmetric key used to encrypt/decrypt sensitive data

The server will maintain a key escrow in order to act as an off-line key distribution authority which does not require on-demand real-time availability of any device. The key escrow also allows the server to pre-encrypt sensitive data before transmission so that device synchronization can occur without requiring a user to be logged on. The key escrow is encrypted using the same key used to protect the server's private key.

In order to avoid having the device accessible with the PIN entered, the institution may specify a time interval of inactivity after which there is an automatic logoff. On logoff, memory is flushed and there are no keys or sensitive data remaining unencrypted on the device.

Passing Data from the Mobile Device to the Server

On connection with the server, the mobile device establishes a secure SSL connection with the server.

All data is moved between the mobile device and the server with two layers of encryption:

- The session key
- SSL

PatientKeeper Data Transport is Highly Defended Against Man-in-the-Middle (MITM) Attacks

A Man-in-the-Middle (MITM) attack is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other. An example of this is when a rogue wireless access point is compromised and used to convince two parties that their communication is secure. The PatientKeeper solution protects against this using a number of mechanisms:

- PK requires HTTPS/TLS communication between device and server
- PHI data is encrypted separately from HTTPS before transmission and is therefore not subject to MITM attacks
- Secondary certificate validation

Secondary certificate validation prevents data exchange over a compromised network. At the time of TLS/SSL communication, the PatientKeeper mobile application obtains the SSL certificate used for communication. The PatientKeeper mobile application then encrypts the SSL certificate and sends it to the server for validation. A validation routine on the server checks the validity of the SSL certificate using PatientKeeper-approved certificate authorities.

If the network is compromised, the PatientKeeper application will not allow further communication. This ensures data encrypted on the devices are always using trusted certificates and there is no tampering with or discovery of keys/credentials in transit.

Securing the Mobile Application

Any communication between the device and server is guaranteed to have been sent from a specific device assigned to a specific user by someone who has knowledge of the user's PIN. PatientKeeper has added additional security to the handheld device. If the wrong password is entered several times, the user is locked out of the system. The number of attempts is configurable by the site administrator, which ensures only authorized users can access data on the handheld. PatientKeeper also utilizes the user password to verify the physician's identity. It is impossible to submit new data without a proper password and this prevents unauthorized data from being uploaded to the hospital information system.

PatientKeeper Platform Layered Security Network

Multiple layers of encryption on the mobile device and server using the latest encryption standards, along with high-speed encryption algorithms and audit trails on the mobile device, server, and backend CDR, ensure that the PatientKeeper Platform exceeds HIPAA requirements for confidentiality of patient data. The Application Server is aware of what data is sensitive and will refuse to transmit those data unencrypted. This is configurable and can be adjusted to meet any hospital's level of HIPAA tolerance.

The PatientKeeper Security Framework assures that, in most cases, the data on a mobile device will have a higher level of security than data on the backend CDR and that data flowing over a wireless network will have more security than data flowing over a local area network.

Support For Industry-Standard Wireless Communication Security Infrastructures

The PatientKeeper Security Framework will coexist seamlessly with industry standard wireless security protocols on Apple iPhone /iPad OS and Android OS. The PatientKeeper Security Framework uses TLS/SSL encryption over a wireless link, combined with a login to the backend clinical data system.

PatientKeeper Platform Brings Secure and Auditable Mobile Computing to the Enterprise

The implementation of the latest federal encryption standards, triple encryption over wireless networks, strong encryption of all HIPAA data on the mobile device, audit trails of all data movement to and from positively identified clinicians, and support for industry standard security protocols allows enterprise security teams to rapidly close any security breach and positively identify the individual responsible for any release of information in a mobile, wireless computing environment. These enterprise controls are essential to maintain privacy and confidentiality of patient data that meet or exceed HIPAA security requirements.

HCA Healthcare℠

Powering HCA Healthcare's physician experience

PatientKeeper®

PatientKeeper, Inc.
880 Winter Street, Suite 300
Waltham, MA 02451

P: 781.373.6100
F: 781.373.6120
www.patientkeeper.com

About PatientKeeper

PatientKeeper® software optimizes healthcare providers' EHR systems to streamline physician workflow, improve care team collaboration, and maximize business and EHR performance. Using PatientKeeper, physicians can easily access and act on all their patient information in a single, intuitive and secure electronic environment – from PCs, smartphones and tablets – fully integrated with a hospital's existing clinical and transactional systems. PatientKeeper is used by more than 70,000 physicians at hospitals and health systems across North America and the UK.

© 2019 PatientKeeper, Inc. All rights reserved. PatientKeeper® and the PatientKeeper logo are registered trademarks of PatientKeeper, Inc.
All other trademarks and brands referenced herein are the properties of their respective holders.

04/19 - PatientKeeper Tech Brief - Mobile Security - 186-688