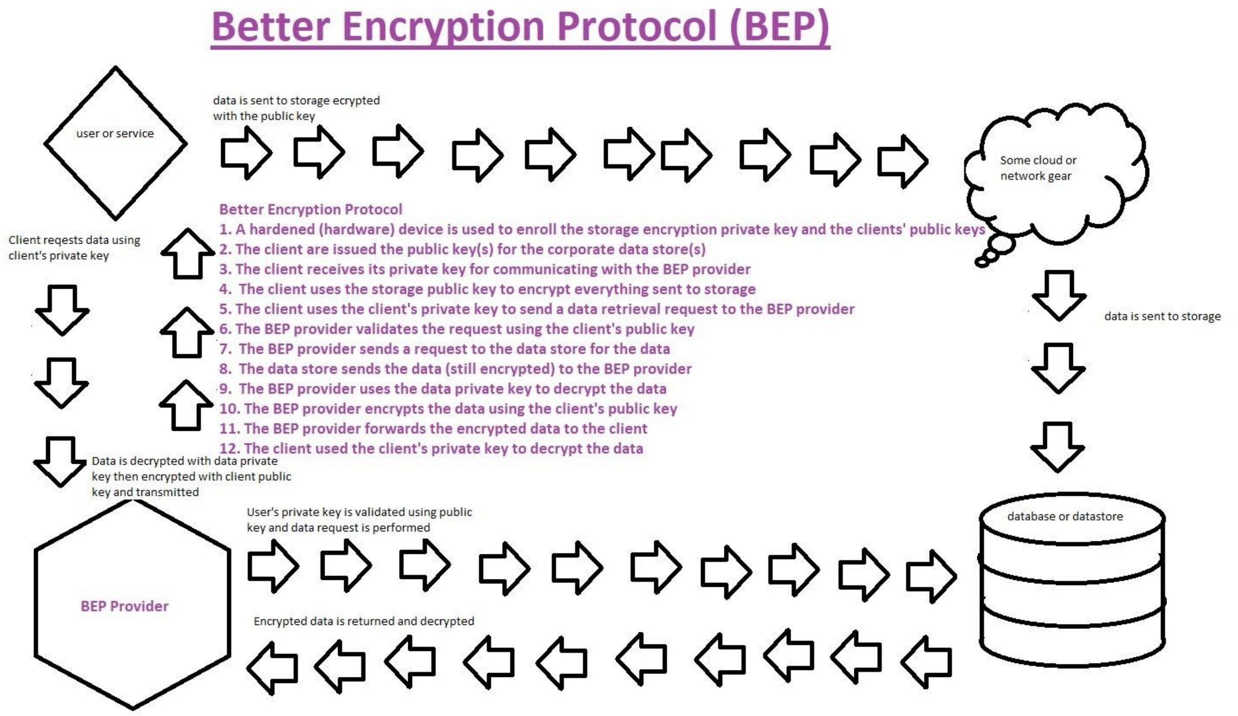

Better Encryption Protocol (BEP)

Better Encryption Protocol

1. A hardened (hardware) device is used to enroll the storage encryption private key and the clients' public keys
2. The client are issued the public key(s) for the corporate data store(s)
3. The client receives its private key for communicating with the BEP provider
4. The client uses the storage public key to encrypt everything sent to storage
5. The client uses the client's private key to send a data retrieval request to the BEP provider
6. The BEP provider validates the request using the client's public key
7. The BEP provider sends a request to the data store for the data
8. The data store sends the data (still encrypted) to the BEP provider
9. The BEP provider uses the data private key to decrypt the data
10. The BEP provider encrypts the data using the client's public key
11. The BEP provider forwards the encrypted data to the client
12. The client used the client's private key to decrypt the data

user or service
data is sent to storage ecrypted with the public key
Some cloud or network gear
Client reqests data using client's private key
data is sent to storage
Data is decrypted with data private key then encrypted with client public key and transmitted
database or datastore
User's private key is validated using public key and data request is performed
BEP Provider
Encrypted data is returned and decrypted

Better Encryption Protocol

Written by Andrew Kosakowski

Table of Contents

Document Introduction

As of the time of publishing this document Better Encryption Protocol (BEP) is a notional protocol that is being released for discussion and consideration for development. This protocol and related documentation is free for all uses, development, implementation, discussion, and/or dissemination as long as this document and this permission-set is attached to all derivative, non-derivative, and/or future works, discussions, development, solutions, implementations, dissemination, and/or all other uses. This document is a technical introduction into Better BEP. The flow is designed to provide a provide an introductory overview of BEP to permit a base layer of understanding that can be leveraged throughout the remainder of the document. Assumption are then provided to provide insights into what was considered to be within the responsibility of BEP and what is the responsibility of other solutions and implementations as well as other relevant assumption. The BEP provider is covered in detail as the first part of the technical portion of this BEP introduction as it is the anchor of the BEP protocol much like a DNS server is the anchor in the DNS protocol. Technical user-side and server-side BEP integration follow to provide insights about how clients and various storage types can leverage BEP based on intended best practices; however, the integration will not be exhaustive as this protocol will require tailoring to specific operating systems, solution providers, and software versions. Once the technical details of BEP is understood three use cases are discussed: databases, file storage, and blob storage. Once some use cases have been considered caveats will need to be discussed followed by ways to automate portions of BEP management. Once some ways to automate portions of BEP certificate management have been provided a wrap-up summary will be provided.

Overview of BEP

BEP is a protocol that has been designed to increase the confidentiality and non-repudiation associated with accessing and updating remote-storage solutions. This is accomplished by using a BEP Provider which is a stand-alone hardware server. This hardware server acts as a remote-storage read-access proxy. Clients (end-users or applications) write directly to the remote-storage using a BEP public key and read from databases via the BEP Provider. The storage solution receives a connection from the BEP provider and clients using standard encrypted storage protocols. Clients connect to the BEP Provider with a client-specific BEP provided client private key. The BEP Provider decrypts storage account data and (if programmed) sensitive storage account data, then re-encrypts the data with the client's public key for transport to the client over HTTPS. Using public-private keys to decrypt data from storage accounts creates non-repudiation while ensuring confidentiality by only permitting the BEP provider has the ability to decrypt data. By using a public-private key pair for encrypting data and writing to storage ensures non-repudiation with data-write, data-save operations, and BEP Provider read requests. Using BEP ensures that any data-base leaks only leak encrypted information that cannot be read and reduces the read attack surface to the BEP Provider and not all ports, protocols, accounts, and clients that have access to the data storage solution.

Assumptions

The assumptions that are known to exist when BEP was being developed include the following:

- Storage solutions will still ensure the integrity of data
- Storage solutions will still use authentication and authorization to secure access access and permissions (which may be limited to only permitting reads from BEP Providers and writes from authorized users)
- Network security will be used to prevent brute-force, DDOS, and other attacks

BEP Provider

The BEP Provider is a security device that acts as the only identity permitted to read from a data storage location and proxy that data to requester identities in a secure fashion that ensures non-repudiation. As we look behind the curtains we see this is facilitated by a more complex set of tasks that include certificate management for the data storage, sensitive data layer encryption, and requester identity authentication and authorization.

> Data storage certificate management by the BEP Provider is accomplished by creating a self-signed asymmetric certificate and publishing the public certificate in a known-location where requester identities can retrieve the public certificate. The private certificate is then internally associated with a data locations. For key-rotation (not-recommended at this time) the BEP provider requires an associated BEP-writter identity that can write all data locations with the new key continuously until all data has been successfully decrypted by the old key by be BEP Provider, then encrypted and written to the storage with the BEP-Writter Identity. Once the New writes have been completed and verified with the new key the BEP-Writter Identity should be destroyed along with its public-private key pair. The reason for using internal BEP Provider certificates instead of purchased public-key certificates is to reduce the attack surface and number of entities that handle the key to one.
>
> Sensitive layer encryption is achieved by mapping specific data with a salt that is added to the data solution keys. As the BEP Provider is the only entity that has access to the private key for the data the salt can be simple, such as the name of the sensitive type of data (ex. “secret” or “SSN”). Once these “salts” are added to the data public key the sensitive data is encrypted then all data (including the sensitive data) is encrypted in entirety by-way-of the unsalted public asymmetric data key. To provide read access to data the BEP Provider decrypts all data with the unsalted private key and provides it to the user identity unless sensitive data is explicitly requested. When sensitive data is explicitly requested the BEP Provider takes the additional steps of verifying the identity is authorized to read the sensitive data type then uses the sensitive data type salts before deliver to the requester identity. In this way, the BEP Provider can provide an extra-layer of data protection to sensitive data in storage by using data-type salts in addition to full data encryption by using the same encryption key.
>
> Requester identities are also managed using authentication and authorization to the BEP Provider using public-private asymmetric key pair certificates. To determine which entities are permitted access to read from the storage solution the BEP Provider requires read access to the organization’s identity provider’s group membership. In particular, the BEP Provider needs to read the list of users contained in a BEP specific group. The BEP Provider reads the group membership daily to determine which certificates to provision and which certificates to revoke. A manual process can be used by the BEP Provider admin account to remove any user certificates that need to be deleted in a shorter time frame. The BEP Provider admin account should be limited to only permit the retrieval of the BEP Provider root certificate for addition to the certificate stores of authorized requester, writer, and storage identities, the removal of read certificates, and a manual read-key rotation.

User-Side BEP Integration

User-side BEP integration consists of adding a BEP salt database, a BEP driver, and the user-specific BEP certificate. When a user wishes to write to the storage provider the BEP driver takes the following steps.

1. Encrypts the sensitive data using the sensitive data salted BEP Public Certificate with the salt found in the local BEP salt database
2. Encrypts all data (including sensitive data) with the unsalted Public BEP Certificate
3. Verifies the certificate with the BEP Provider
   a) If the BEP Driver fails to connect with the BEP Provider it makes 3 additional connection attempts before failing with a user pop-up error box
   b) If the BEP Provider does not verify the certificate as legitimate the BEP driver retrieves the authorized public certificate and begins from step one. This order of operations ensures only legitimate data is written to the data solution using the legitimate key
4. Establishes an encrypted connection to the data solution
5. Transmits the write commands and encrypted data to the database

Storage-Side BEP Integration

The storage-side BEP integration is limited to removing all read access except for management admins that have operation need to perform administrative tasks on the data solution and the BEP Provider. The BEP Provider should be configured to use the enterprise specific authentication mechanisms that are already in place. This could be any identity provider or the storage solution itself (in the case of databases or solutions that do not integrate with enterprise identity providers). The BEP Provider identity should be restricted from any access that is not required to read the entirety of data in the solution.

Use Cases

Three use cases will be discussed to provide the greatest variety of context and applicability of BEP to meet the needs of the widest array of industries and requirements.

Data Bases

Using BEP in conjunction with a database data solution can be leveraged to ensure the highest degree of non-repudiation and leak resistance. This is an ideal scenario where databases have mixed sensitivity content. Data of various sensitivity levels can be stored in a mixed sensitivity databases while ensuring confidentiality by using the sensitivity salts. In this scenario sensitivity-specific identities (such as a top-secret, secret, and confidential workstation) could all write to the same data base and because each workstation only has access to the salts at their level and no access to the private 'read' key confidentiality is maintained. A good example of this type of use is a mixed HR/Finance database where all data from both departments is written to a single database and only finance identities have access to finance related columns (such as bank account specific details) while HR only has access to HR specific columns (such as home address).

File Shares

Employees love to share data and save files with ease and without being burdened by security requirements or red-tape. BEP can assist with this by being leveraged to ensure personal file drives are only accessible by the employees to whom the drives are assigned. Likewise, for group projects, directories can be mounted as drives permitting a directory on a file share to be maintained using BEP. In both use cases the data can be configured to be read from or written to by one or many users using the identity provider groups identified in the BEP Provider section. To add the sensitive data encryption layer a specific directory, file type, or file tag can be used to identify sensitive files but this functionality is not currently part of the nativly suggested BEP implementation.

Blob Storage

Blob integration with BEP as is a merger between the database and file share scenarios. In this scenario BEP is used to encrypt all data in blob storage while providing sensitive storage encryption to specific blob containers. For instance, all enterprise data could (in theory) be stored in a single blob storage location. If that blob storage were broken down into a “top-secret”, “secret”, “confidential”, and “public” container the BEP Provider can use salted keys for the sensitive storage containers and much like in the database scenario only authorized identities could have knowledge of and authorization to those sensitive containers.

Automation

There are a number of places where BEP can utilize automation for easier management and integration into any enterprise environment. The first place where automation can be leveraged is in the authorization of reader identities. By using dynamic group membership in the BEP Provider specific groups the BEP Provider can automate certificate creation and deprovisioning. Once read access authentication is automated the rotation of data solution keys (the write keys) can be automated based on a defined schedule. While the key is rotated the database should be locked for write actions (or a duplicate database should have a key rotated while the primary is written to, then the databases synch). The final place where BEP can be automated is the sensitive label salts. The BEP Provider can leverage sensitivity specific groups in the identity provider to discover and integrate salts into a sensitivity layer encryption and issuance to authorized data readers.

Caveats

Some caveats to consider are the processing power need to decrypt and encrypt data inline. Current processing power options may not be sufficient for heavy workloads and may require using a load balance

Summary