Hewlett Packard Enterprise

The Identity-Based Encryption Advantage

Contents

Technical white paper

Introduction

This technical brief will define the requirements for a robust key management system, explain why traditional key management architectures do not fully meet these requirements, and introduce a new architecture that uniquely meets all the requirements for an effective enterprise key management system.

Six Requirements for Enterprise Key Management

An enterprise key management system needs to perform multiple jobs. In an enterprise environment, whether protecting email, databases, or documents, a key management system must handle the following six (6) requirements:

- **Requirement 1:** Deliver encryption keys. Users and applications must be able to encrypt data to all the recipients needed to keep business processes functioning. If a user can't specify a wide variety of internal recipients, directory groups, customers, and partners, it may be impossible to encrypt that data. If a user cannot easily encrypt the data, the business process stops functioning or, worse, the user will store the data unprotected and in the clear.
- **Requirement 2:** Authenticate users and deliver decryption keys. The key management system must be able to both authenticate and deliver keys to the groups and users specified by the data sender. The authentication system should integrate with existing infrastructure, or the infrastructural and operational costs of authenticating users will be duplicated.
- **Requirement 3:** Jointly manage keys with partners. In many cases, it is desirable to exchange encrypted data with partners, where each partner manages only the keys for its own users. In this way, a message from Company A can go to recipients in Company B without requiring that Company A knows how to authenticate Company B's users. Company B can manage its own authentication and message handling infrastructure without needing to coordinate with Company A.
- **Requirement 4:** Deliver keys to trusted infrastructure components. When encrypting email or files in an enterprise, intermediate business and technical processes, such as auditing, content scanning, or anti-virus, can require access to the contents of encrypted data. The key management system must be capable of delivering keys to these components.
- **Requirement 5**: Recover keys. In the case of a user leaving the organization, machines losing disk storage, or any number of other cases, an administrator must be able to access encrypted data. Recent e-discovery rules have made this of paramount importance. If a key management system leaves a data item in a state where it cannot be decrypted, this can lead to significant legal liability and a disruption in business processes.
- **Requirement 6:** Scale for growth. Enterprises are not in business to shrink. Successful businesses grow, and sometimes, they do so quite rapidly. An effective key management system must accommodate a large transaction volume and must be deployable in load balanced and geographically dispersed configurations. Even if this is not a requirement for a specific application today, companies must consider how key management will work if, and when, they encounter 10 or even 1,000 times the transaction volume they currently handle.

Let's review the two most prevalent traditional key management systems, how they function, and how they fall short of meeting the six requirements of an effective enterprise key management system.

Traditional Approaches to Key Management

Symmetric Key Management

The first and oldest—dating back to the 1970s—dedicated key management architecture uses the same data encryption technology to manage keys and scramble data. In these systems, called "symmetric key" systems (Figure 1), because the same key is used to encrypt and decrypt information, the key manager generates a new key for every message at the sender's request. The key is stored in a database along with the list of receivers. When the receiver authenticates, the key is retrieved from the database and the receiver name is matched against the list of authorized recipients. If everything checks out, the decryption key is sent to the receiver.

Symmetric key systems have become the centerpiece of internal-only encryption and authentication systems. Until recently, Kerberos systems and Windows domain controllers were based around symmetric key management. The ability to translate passwords readily into keys and the extremely fast performance of symmetric key encryption algorithms make these systems attractive for internal applications that do not need to include any external users in the encryption process.

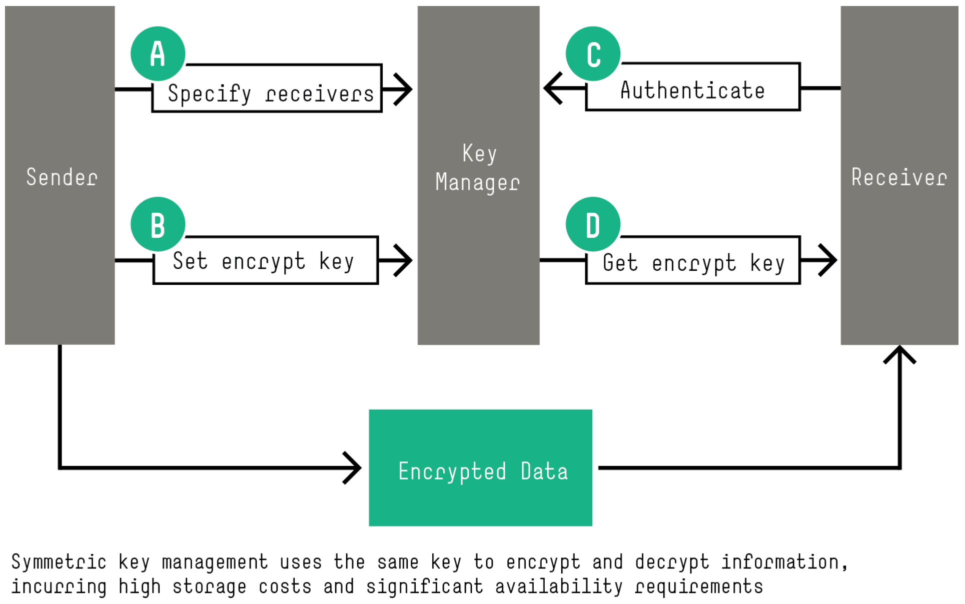

Sender
A
Specify receivers
B
Set encrypt key
Key Manager
C
Authenticate
D
Get encrypt key
Receiver
Encrypted Data
Symmetric key management uses the same key to encrypt and decrypt information, incurring high storage costs and significant availability requirements

Figure 1. Symmetric key management diagram

High Storage Costs—Many, but not all, symmetric key systems require that a database containing the key for every message is present in the system. While some proponents of symmetric key systems will insist that this database is not a significant impediment, this key database must be replicated, backed up, and generally managed. Because this database holds security critical information (namely, the keys), all of these costs are magnified.

High Availability Needs—Because the sender must request a key for each message from the key manager, the key manager is involved in every encryption operation. This means that the key manager must be highly available and the scale of the key manager will limit the scale of the entire messaging or data encryption system. This also tends to complicate the repercussions of the storage needs.

Shortcomings of Symmetric Key Management

While they do not dramatically affect the delivery of encryption keys (Requirement 1) or the authentication of users and delivery of decryption keys (Requirement 2), the storage and availability needs of symmetric systems do significantly impact the remaining four requirements Here's how symmetric key management falls short of meeting all six of the requirements of an effective enterprise key management solution:

- **Requirement 1:** Due to its simplicity, symmetric key systems can be used to encrypt data to any recipient. Connection to the key server is required for every unique encryption, however, since a unique key is required for each operation and it must be shared with the server. This also means that offline operation is not possible, and the volume of connections is quite high.
- **Requirement 2:** As with encryption, decryption requires that a connection to the key server be made for each and every transaction. This precludes offline operation and has the potential to negatively impact scalability.
- **Requirement 3:** Because senders must request per-message keys, if a sender wishes to make a key usable to a partner, the sender must be able to contact the partner's key server for every message. This means that partner key servers must always be online and available on the Internet, creating significant security risks and scalability problems.
- **Requirement 4:** While symmetric key systems can deliver keys to trusted infrastructure components, these components will need to request a key per each message processed. For example, if an email appliance scans all incoming messages, it will need one key for message examined, imposing a significant load on the key server. Any infrastructure that sends automated messages will also need to request encryption keys.
- **Requirement 5:** Symmetric key systems can recover keys as long as the key database can be recovered. If the symmetric key database is damaged or lost, the ability to recover keys will be similarly impaired.
- **Requirement 6:** Scalability of symmetric systems is limited by the per-encryption requirement to contact the key server as well as by the ability to grow and manage the key database.

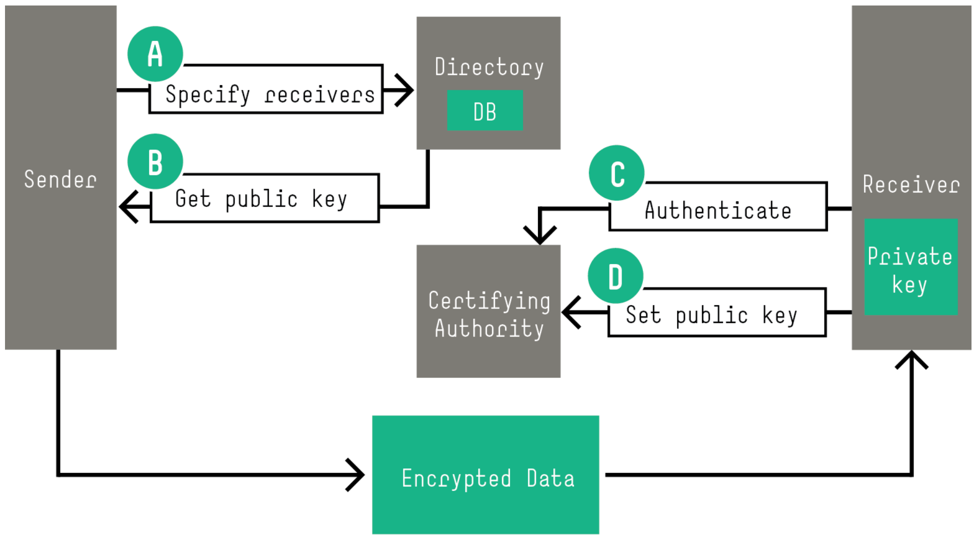
Public Key Infrastructure (PKI) Key Management

In the early 1980s, a series of mathematical innovations led to important new kinds of encryption algorithms. These algorithms, called "public key" or "asymmetric" systems use a different key to encrypt data than the one they use to decrypt data. The famous Diffie-Hellman and RSA algorithms are the best-known examples of public key algorithms. While they are ill suited to encrypting large chunks of data, asymmetric algorithms are perfectly suited to managing keys, since they can readily encrypt smaller, key-sized objects.

The essential idea behind using public key algorithms to manage bulk encryption keys is that a recipient generates a pair of keys: one public key and one private key. To encrypt data, the sender generates a bulk encryption key, encrypts the bulk key with the recipient's public key, and sends the data along with the newly encrypted bulk key. The recipient gets the data, decrypts the bulk key with his private key, and then uses that key to decrypt the data.

On the surface, systems based on the public key infrastructure, or PKI, model seem to solve the most pressing flaws of symmetric key systems: there is no need for a per-message key database and the key server does not need to be contacted for each message. In reality, however, PKI has two significant disadvantages with respect to its predecessor, symmetric key management: 1) generating the private key at the recipient makes key recovery difficult to implement and 2) the sender must locate a public key for each recipient and verify its validity. Since the recipients generate these keys themselves, the server may not be able to supply keys for all recipients.

This inability to find keys for every recipient, made key management for encrypted email systems, such as PGP and S/MIME, impractical for encryption. This is the problem certificates were intended to solve. Certificates are authenticated data structures that tie a receiver's identity to a public key, without the need to request the key from a server. Because they are authenticated, certificates can be stored on distributed, untrusted directories, rather than the key server. This splits the key management server into a public facing directory and a Certifying Authority (CA). The CA (Figure 2) is the only trusted component and the untrusted directory handles most of the transactions in the system, allowing for easier scalability.



PKI key management systems do not require a per-message key database or contact with the key server for each message, but they are impractical to use and make key recovery difficult.

Figure 2. Certificate-based key management diagram

Shortcomings of PKI Key Management

How do PKI-based systems up against the list of six requirements for an effective enterprise key management system? Interestingly, the more advanced technology actually has a greater number of shortcomings than the prior generation:

- **Requirement 1:** Public key-based systems have the inherent problem of locating a public key for a specific user. Users of certificate-based systems often find that they cannot locate a certificate for a specific user, making them unable to encrypt the data. Groups present a similar problem, because implementing certificates for groups requires a private key distribution scheme and the authentication scheme for the certificates must accommodate moving users in and out of groups.
- **Requirement 2:** Typically, user authentication in a certificate-based system must happen before the user can receive a message. This means that there must be an enrollment step that takes place before data can be encrypted to a receiver. In the case of customer or partner facing encryption systems, this may be an insurmountable flaw. Assuming this step has been overcome, the receiver can decrypt the data as long as the corresponding private key is available.
- **Requirement 3:** Partner-to-partner communications can be done in a public key-based system, but a shared directory must be exposed.
- **Requirement 4:** To deliver keys to an intermediate scanner or archive system, a private key sharing system must be deployed or a policy set to encrypt the data item to the intermediate system.
- **Requirement 5:** Keys can be recovered if a private key database is maintained. If that database is compromised or damaged, keys can become unrecoverable. Clients can be programmed to encrypt data to a trusted intermediate, which allows for messages to be recovered at that intermediate.
- **Requirement 6:** Theoretically, public key systems can scale. However, experience with large deployments has shown that the user interface issues with respect to managing and revoking certificates and keys make wide-scale use of PKI very expensive.

REQUIREMENT	SYMMETRIC KEY MANAGEMENT	PKI
1. Encrypt	Yes, online connection required.	Often no, when no recipient certificate is available.
2. Decrypt	Yes, online connection required.	Yes.
3. Manage with partner	Yes, but must perform per encryption connection.	Yes, but must publish a directory externally.
4. Integration with infrastructure	Yes, but requires a per decryption lookup.	Not without complex key escrow and sharing.
5. Key recovery	Must maintain a key database.	Must maintain a key database.
6. Scalability	Limited by per-transaction key server operations.	Limited by operational complexity.

Both symmetric key management and PKI fall short of meeting all six of the requirements of an effective enterprise key management system.

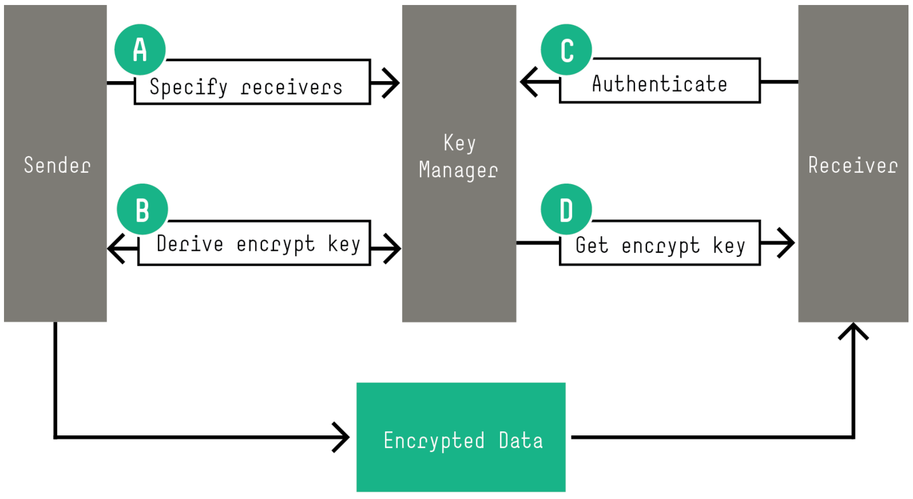
A New Approach to Key Management

Adi Shamir, one of the pioneers of public key cryptography, proposed a new type of public key algorithm in 1984. While public key systems have the inherent problem of distributing public keys and tying those public keys to a specific receiver, Shamir proposed mathematically generating the receiver's public key from his/her identity, then having the key server calculate the required private key. This system, called an "Identity-Based Encryption" (IBE) algorithm, would remove the need for public key queries or certificates. Because the key server generates the private key, key recovery no longer requires a separate private key database.

In 2001, Dan Boneh and Matt Franklin solved the mathematical problem and constructed the first practical and secure IBE, now HPE Identity-Based Encryption (IBE) system.

Using IBE (Figure 3) radically simplifies key management because the sender does not need to contact the key server to get an encryption key. Instead, the encryption key is mathematically derived from the receiver's identity. The receiver must only contact the key server once to authenticate and get the required decryption key. The key server is able to construct the receiver's decryption key mathematically, eliminating the need for a database at the key server and making key recovery extremely straightforward.

Since the sender does not need to contact the key server per unique recipient, encrypting information on a partner's key server is simple. The sender's policy can dictate which key server will be used to protect a message. That server can be one controlled by the sender's organization, one run by an outside service, or one located at the receiver's organization.



IBE key management removes the need to communicate with a key server in an encryption transaction.

Figure 3. IBE diagram

How IBE Uniquely Meets the Six Requirements of an Effective Key Management System

These technical innovations translate into tangible benefits when looking at the six requirements for an effective enterprise key management system. As encryption migrates from a server-to server and internal-only application to an information encryption utility, IBE becomes a natural choice for managing encryption keys because it is the only architecture that meets all six requirements of an effective key management system:

- **Requirement 1:** Deliver encryption keys. Using IBE, keys are always available for all recipients. The encryption key is derived mathematically from the receiver's identity. Groups are handled just as easily, because a key can be made from a group name as easily as an individual name.
- **Requirement 2:** Authenticate users and deliver decryption keys. IBE interfaces with existing authentication infrastructures, so any authentication resources that are already deployed (e.g., directories or web authentication) can be reused. The customer experience for getting a decryption key can be the same as logging into a portal, for example.
- **Requirement 3:** Jointly manage keys with partners. IBE enables the sender to select a local key server, a partner's key server, or a service to protect the data, depending on its particular requirements.
- **Requirement 4:** Deliver keys to trusted infrastructure components. Because IBE mathematically generates all keys at the server, the server can securely regenerate keys for infrastructure components as needed.
- **Requirement 5:** Recover keys. In an IBE-based system, all keys are generated from a base secret stored at the key server. This base secret is backed up at server generation and, as long as the secret can be retrieved, any key can be securely regenerated.
- **Requirement 6:** Scale for growth. Without the need for databases that grow over time or requirements for per-transaction connections to the key server, IBE enables additional applications and transactions to be added with very little, if any, additional key management infrastructure. Key servers can operate independently, allowing for geographic dispersion and load balancing.

Because IBE represents a genuine breakthrough in managing keys for encryption, it enables encryption to become a mechanism to universally enforce access control at the document level. How IBE enables data-level security can be seen by examining a common application for encryption: protecting email. Other applications, such as file encryption and application-level data protection, will function similarly.

HPE Identity-Based Encryption (IBE) uniquely meets all six of the requirements of an effective enterprise key management system.

REQUIREMENT	SYMMETRIC KEY MANAGEMENT	PKI	HPE IDENTITY-BASED ENCRYPTION (IBE)
1. Encrypt	Yes, online connection required.	Often no, when no recipient certificate is available.	Yes, to anyone including groups.
2. Decrypt	Yes, online connection required.	Yes.	Yes, without pre-enrollment required, including offline support.
3. Manage with partner	Yes, but must perform per encryption connection.	Yes, but must publish a directory externally.	Yes.
4. Integration with infrastructure	Yes, but requires a perdecryption lookup.	Not without complex key escrow and sharing.	Yes. No per message lookup or key escrow required.
5. Key recovery	Must maintain a key database.	Must maintain a key database.	Yes. No database required.
6. Scalability	Limited by per-transaction key server operations.	Limited by operational complexity.	Yes.

IBE Applied: Key Management for Encrypted E-mail

How does key management impact encrypted email systems? Is key management actually worth worrying about when evaluating a system to secure email traffic to customers and partners? The answer is an emphatic "YES". In fact, the underlying key management system is arguably the most important factor governing the performance of a secure email system and, more importantly, the quality of the experience for end-users.

The simplest way to understand how key management impacts the overall user experience is to open an email client and attempt to send a secure email to a customer or partner using S/MIME, a mature technology built into the most widely deployed email clients on the market today. Tellingly, 99 percent of the time, the user will fail at the very first step: finding a certificate for the recipient. Why? Because while S/MIME has been around for years, S/MIME messages are so rare that they may as well be nonexistent. This failure to conquer the market is not due to poor software engineering. Hundreds of implementations exist and numerous attempts to fix these fundamental limitations have all failed. The problem is inherent in the key management layer: getting users to enroll for certificates and distributing those certificates once users are enrolled is just too difficult.

In contrast, using IBE, secure messages can be sent to any recipient, without first requiring the recipient to take any special action. The ability to send is inherent in the system, and all usability benefits flow from this critical difference. Once the recipient receives a secure email, a simple connection to the appropriate key server to authenticate and receive the decryption key is the only step necessary before he or she can successfully access the secure message.

Behind the scenes, IBE does not need or store user keys or certificates, thus incurring minimal operational overhead. Support for requirements 3 through 6 then follows from the automatic and transparent key generation and secure regeneration. The result? An effective enterprise key management system that not only enables simple, secure encryption and decryption, but also makes these functions manageable, scalable, and user-friendly.

IBE Industry Standards and Global Adoption

IBE is standardized under a number of globally recognized standards including:

- IETF 5091 "Supersingular Curve Implementations of the BF and BB1 Cryptosystems".
- IETC 5408 "Identity-Based Encryption Architecture and Supporting Data Structures".
- IETF 5409 "Using the Boneh-Franklin and Boneh-Boyen Identity-Based Encryption Algorithms with the Cryptographic Message Syntax (CMS)".
- IEEE 1363.3 "IEEE Standard for Identity-Based Cryptographic Techniques using Pairings".
- ISO 15946-1 "Pairings Based Crypto (2008)".

IETF is the Internet standards body recognized globally. IEEE 1363.3 is the global home of public key encryption standards including Elliptic Curve, RSA, Diffie Hellman and Discrete Log Cryptography. ISO is a top European standards body. IBE standards have been adopted across industry, including national health schemes in the EU, and continues to see rapid acceptance across the world.

Today, IBE is in use daily by hundreds of millions of users world-wide by thousands of enterprises and their customers for securing email, mobile data, files, cloud and payment transactions. IBE technology powers billions of key operations easily and efficiently every year.

IBE Technology powers products from HPE Security—Data Security as well as leading industry service, cloud and software providers including 7 of 10 top US Banks, 6 of 8 top US Payment processors, and systems powering banks, retailers, insurance companies, healthcare providers and government organizations.

Conclusion

Traditional approaches to key management, including symmetric key management and, more recently, PKI, have fallen short in meeting the six requirements of an effective enterprise key management system. Rather, they are difficult to use, implement, and manage, and extremely expensive as well. In contrast, HPE Identity-Based Encryption, or HPE IBE, uniquely meets all six requirements by encrypting data, authenticating users and decrypting data, jointly managing keys with partners, delivering keys to trusted infrastructure components, recovering keys, and scaling for future growth. Plus, HPE IBE meets these requirements in a cost-effective and user-accessible manner, ensuring adoption and, ultimately, the security of electronic communications.

4AA6-0086ENW, April 2016, Rev. 1