sertainty

data: empowered

Self-Protecting-Data vs. Standard File Encryption

THE THREE DIFFERENTIATORS

“We have to move to where our data is far more **aware** and where our data is essentially helping to **protect itself** – so that it knows where it is, who’s trying to access it, and a lot of context around it so that it can be protected – whether it’s on a computer that gets lost in a parking lot or left on an airplane or someplace else that it is not secure.”

GRANT SCHNEIDER

US FEDERAL CISO

A Much-Needed Breakthrough in Data Security

In a 2018 panel interview, Grant Schneider, US Federal Chief Information Security Officer and the National Security Council's Senior Director for Cybersecurity Policy, articulated the need for security at the data level itself. Note how he uses the term "aware" to describe how secure data might behave in context-sensitive ways to protect itself.

Thomas Sasala takes it a step further. Sasala, Chief Data Officer for the U.S. Navy, articulates both why we need data-level security, and what it might mean for user access.

- "The adversaries are not stealing our network; they're stealing the data on the network."
- "[If] the data isn't protected at the data level — not at the perimeter level or even at the server, system or application level — then we're not going to actually survive moving into the future."
- "Data at rest, data in transit, data encryption: these things all need to be tied together…"

Sertainty offers breakthrough technology in that it gives data the awareness to act and react, creating a new level of protection at the data layer.

Sertainty Self-Protecting-Data contains a number of advanced protection features that set it apart from standard encryption products and key management methods. This paper provides a quick explanation of these features, how they reduce the complexity and burden associated with the management of data protection, and how Sertainty Self-Protecting-Data provides a depth of data-centric protection that standard encryption products alone cannot attain.

Encryption Product Basics

The basic tenet for all encryption products today is the encoding of a message or information so only authorized parties can access it, and those who are not authorized cannot. When data access is desired, the data will be decrypted and written back into its original format. All data encryption products utilize keys for encryption. These keys require protection, outside of the encryption application itself, including user authentication and authorization to determine if the user has the necessary privileges to access the data in the encrypted file.

The two primary methods of data encryption are: Symmetric-Key Encryption and Public-Key Infrastructure (PKI or Public-Private Encryption).

Symmetric Key Encryption is the use of a singular encryption key for both encryption and decryption of sensitive data. This key can easily be visualized as resembling a door lock key, in that the same key is used for protection as well as access, and must be protected, but shared if others are to gain access to the protected data or area. You wouldn't leave your door key hanging on a hook next to the door! The symmetric key/lock example also demonstrates that external actions must still be taken to protect the keys, and hence the data, from theft. These protections include authentication, authorization (knowing who possesses the key and that they have permission to use it), and secure sharing of the key. Still, this provides access to the entire file for decryption. There are no controls beyond being allowed access to the data or not being allowed access to the data. If lost, the locks/keys must be replaced and securely re-distributed.

Public-Private or PKI Encryption provides for a key pair that allows one key to be used to encrypt data (example: the recipient's public key which is published or publicly shared), and a matching key (the recipient's private key, that is kept private) that is the only key capable of decrypting the public-key encrypted data. This encryption/decryption process can be operated in either direction, public encryption-private decryption/private encryption-public decryption, and through the simultaneous use of both sender and receiver's key pairs, i.e. encrypting data with the public key of the recipient guarantees only the recipient, who holds the private key can decrypt the data, and is signed by the private key of the sender, so the recipient can, by being in possession of the sender's public key, guarantee the sender.

This PKI-enhanced process adds:

- Authentication – The sender knows the receiver, and the receiver knows the sender
- Non-repudiation – The sender cannot deny the transaction
- Data integrity – Through data hashing of the original message, the integrity can also be determined

Encryption Product Basics Continued

Both Symmetric-Key and PKI Encryption require external systems to provide management of the keys. In Symmetric Key Encryption, the only function provided by the key is protecting the data. All additional functions, such as authentication, authorization, etc., are managed externally. In Public-Private Key Encryption, authentication/authorization and non-repudiation are inherent in the transaction, but require a more complex environment, as PKI is not natively built into many operating systems or applications and is primarily used for data-in-transit security requirements. Keys also expire. So, it is the function of the key management system to provide keys on request and expire keys when their validity ends.

Neither of these encryption methods monitor for compliance, nor do they contain behavioral responses to unmet policy requirements, or self-logging of activity. Both encryption methods also require the data be decrypted completely for use when changing the data state from at-rest to in-transit. They also leave fully-decrypted/unprotected data files written to disk and require access to external (to the data) key management solutions for encryption and decryption.

In short, data encrypted using standard encryption methods can only be protected and has no ability to manage or protect itself!

The Challenges of Key Management

The heart of all data encryption methods, tools, and products are the encryption keys. These keys must be managed in accordance with best management practices.

"The proper management of cryptographic keys is essential to the effective use of cryptography for security. Keys are analogous to the combination of a safe. If a safe combination is known to an adversary, the strongest safe provides no security against penetration. Similarly, poor key management may easily compromise strong algorithms."
NIST Publication 800-57 Part 1, Revision 4

Not unlike a combination for a safe, encryption keys are only as good as the security used to protect them and provide unfettered access to them when required. Additionally, there is a full key lifecycle, which must include:

- Key generation, pre-activation, activation, expiration, post-activation, escrow, and destruction
- Physical and/or logical access to the key server(s) for key lifecycle management
- User/Role access to the key(s), including user identification, authentication, and authorization policies of access

Adding to the complexity of the key management lifecycle, different types of keys are used for different purposes:

- Symmetric keys are used in stream cipher mode for encrypting data transfers, or in block cipher mode for encrypting data at rest. Keys are shared between data source and destination.
- Public/private key pairs are used for protecting data in-transit/movement. Public keys are either resident in a directory, or shared individually, and stored in a keyring. Private keys must be kept secure, yet available. In today's proliferating BYOD environment, this is quite a challenge.

Additionally, each key management server type, including web-servers, VPNs, SSH servers, secure transport servers, HSMs, etc., all have their own discreet method of key lifecycle management and key updating.

The Challenges of Key Management Continued

As the data is moved from a protected data store across a transport method to a destination, it's protection often needs to change to match the discreet encryption methods supported by the sender, transport, and receiver. From encrypted data on an originator's disk, data must be unencrypted, converted to a message protected with a shared symmetric key or public/private keypair, then decrypted and re-encrypted to be written into the recipient's store. It's not unusual for the data to be unencrypted/re-encrypted multiple times for a single transfer.

A host of products and platforms have been created whose sole job is to attempt to help simplify management of the encryption keys for both users and enterprises, yet this has caused enterprise key management to become a complex task.

Per a February 2019 Egress Data Policy Survey, barely one in five organizations implement encryption policies while sharing sensitive data intra-system, and only one in three implement encryption policies while sharing sensitive data inter-system.

To summarize, external key management inherently creates a time-intensive, costly, complex, difficult to manage environment, and provides a method of attack, both through attack of the credentials that authorize access to the keys, and through denial-of-service.

Differentiator 1: Sertainty Serverless Key Management

All encryption products require the recipient to have access to the decryption keys. See Figure 1. Sertainty SPFiles are data files that are self-aware and self-protecting. The need for an external key creation, management, and expiration is removed - as the keys used for SPFile encryption are dynamically-created, single-use symmetric keys created by the Sertainty solution during the process of creating the SPFile. These keys, along with the SPFile identities and policies, are securely protected inside the SPFile itself through the embedded Sertainty Intelligence Module. The Intelligence Module contains and enforces policies for user access, access limitations including date/time, machine/network access, data expiration, self-logging, and mitigating actions for non-compliance, up to and including data shredding. Upon receipt of the data, the recipient, authorized by the embedded policy to access the data, will then be able to unprotect the data and expire the keys; securely store the data until the expiration date is reached; or the file is deleted. This creates a secure, encrypted data file that has no key or management server or process requirement. See Figure 2. This also creates a single, secure storage and transfer mechanism for use in sensitive environments where external key management servers are not available or reachable. It also protects against a bad actor from performing attacks on the key management server, potentially denying access to keys and their protected files, or from performing other access attempts and potential compromise.

Additionally, as the policies for access are contained within the file, there is no authentication/authorization server that can be monitored or attacked for access to the encryption keys.

Many companies have simplified their key management challenges, reducing risk and cost through the use of the Sertainty SPFiles Platform, including those in the finance, data-science, user authentication, media, healthcare, and government fields.

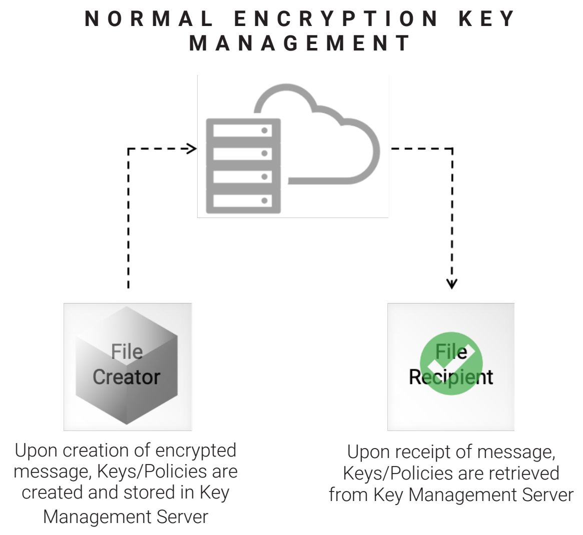

NORMAL ENCRYPTION KEY MANAGEMENT
File Creator
File Recipient
Upon creation of encrypted message, Keys/Policies are created and stored in Key Management Server
Upon receipt of message, Keys/Policies are retrieved from Key Management Server

FIG 1.

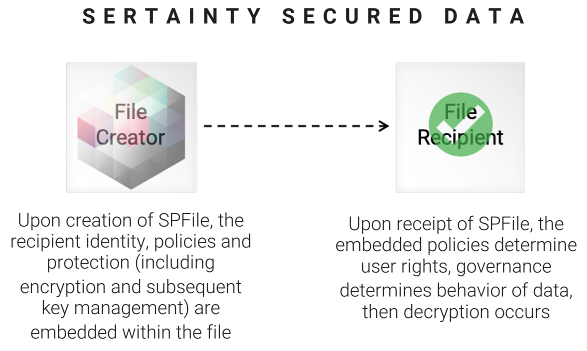

SERTAINTY SECURED DATA
File Creator
File Recipient
Upon creation of SPFile, the recipient identity, policies and protection (including encryption and subsequent key management) are embedded within the file
Upon receipt of SPFile, the embedded policies determine user rights, governance determines behavior of data, then decryption occurs

FIG 2.

Differentiator 2: Sertainty Selective Data Decryption

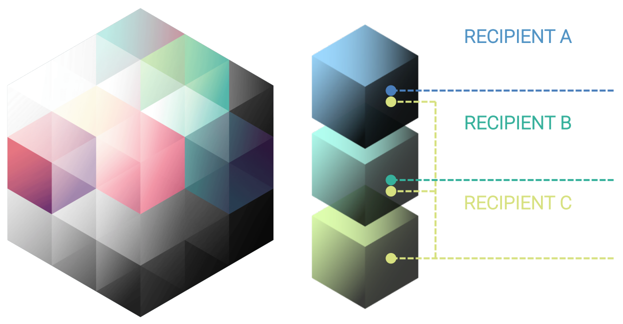
With current, standard encryption methodologies, for a recipient person/service/etc. to access and utilize the encrypted data in a protected file, the file must be fully decrypted. This process takes time, and results in a previously encrypted file now written unencrypted to storage, which is often not removed afterwards. This increases the risk that the file can be accessed in its unprotected state at a later time. Sertainty mitigates this risk and shortens the decryption process time by allowing decrypting/unprotecting of only a part of the data file, a single file in a folder, or a group of files. Ultimately, Sertainty provides selective access to the decrypted portion of the data, while maintaining full encryption protection for the remainder of the data.

As a Sertainty SPFile is created, the original data file is broken into an arbitrary number of individual segments. Each of these data segments are then protected by its own dynamically-generated AES-256 bit key. While this is part of the security aspect of the self-protection features of the Sertainty SPFile, it also adds a unique secure-access feature. This feature allows each segment to be decrypted according to policy via the embedded Sertainty Intelligence Module. This effects unprotected access to only a portion of the file/folder/disk by an individual policy, user, or process. See Figure 3.

Sertainty customers have reported significantly increased processing speed and security enhancements on process-intensive applications through the use of this feature. For example:

- A process accessing a Sertainty SPFile will not need to fully decrypt the file, act on the data, then re-encrypt the file. This allows faster access to the file by process and does not leave remnants of decrypted data in memory or on disk.
- A recipient user may receive a SPFile that contains many files (including folder structures, directories, data disks, sandboxes, etc.), but by policy they are allowed access to only a controlled subset of these files or portion of a single file. They will not see or have access to the other files, and therefore cannot unprotect and expose them.

SELECTIVE DATA DECRYPTION/ACCESS



With selective decryption, the embedded policies can grant users or processes can be granted access to a select portion of the data file, speeding up the process of decryption, and preserving security for the rest of the data in file

FIG 3.

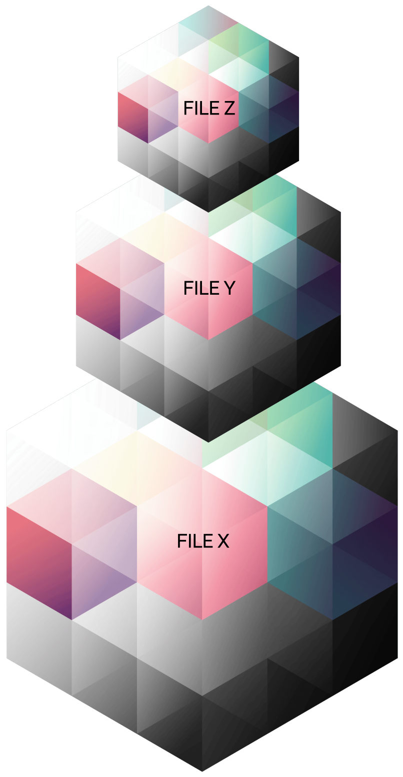
Differentiator 3: Sertainty Secure-Data File Policy Nesting

Sertainty SPFiles can be nested within another Sertainty SPFile, with each file containing unique data and policies, to allow creation and enforcement of a chain of file recipient execution. With standard encryption tools, files can be nested within other encrypted files with different keys. However, a recipient of a Sertainty SPFile has no need to contact the key manager individually for rights to decrypt. Each SPFile contains its own policies above and beyond "decrypt" only (such as time/location/reaction policy behaviors). A SPFile does not have to be decrypted so the next nested encrypted file can be reattached to a message and re-sent. Further, Sertainty nested SPFiles can be protected using selective data decryption to prevent simple encrypted files from being written to disk and left for discovery and potential attack. See Figure 4.

Complex Contract SPFile Nesting Example: A Primary Contractor creates a set of engineering documents for a new building. Document X is a primary design document. Document Y is an interior build-out design plan. Document Z is a materials list. These documents need to be reviewed, modified as needed, and approved in order by the Construction Sub-Contractor, the Build-Out Sub-Contractor, and finally, the Materials Supplier. A nested SPFile can, by policy, control the order in which these files are opened, by whom, what content is available to them for examination or modification, and what policy enforcement method takes place.

For example, the Primary Design Document is sent to the Construction Sub-Contractor for modifications, approval, and sign-off. Once it's approved, the Interior Build-Out Design document, with any new changes/modifications, can be approved and signed off by both sub-contractors. Once the design documents for the external building and internal build-out are approved, the Materials List, can then be opened and modified as needed by both Sub-Contractors, and sent to the Materials Supplier for approval and fulfillment. Further, the Design Documents must be signed during working hours so that assistance may be provided if necessary, while the Materials List must be signed before an expiration date.

SECURE FILE POLICY NESTING



File X, contains File Y and File Z, each with their own policies and user IDs. File Y contains File Z. File X is unprotected to expose File Y which is unprotected to expose File Z. Each file can have discreet policies that determine protection, time, user, etc.

FIG 4.

Summary

For organizations who seek to assure the integrity of their sensitive digital assets and compliance by their processes, users, and supply-chain, Sertainty is the only total data-layer protection and policy enforcement tool that embeds intelligence directly within data files. Unlike network, infrastructure and key management-dependent solution providers, the Sertainty Self-Protecting-Data Technology Platform easily integrates with legacy systems, and allows organizations to automate real-time data policy enforcement, protection and monitoring capabilities, transforming data from a liability to an asset.

To learn more about Sertainty Self-Protecting-Data contact us today!

sertainty.com | sales@sertainty.com

sertainty
data: empowered