

Cryptography as a Service

By Jeff Stapleton – ISSA member, Texas, USA

The security attributes for protecting data in a cloud application environment are discussed, and the ramifications for using cryptography in a cloud environment are explored.

Cloud computing is a conceptual architecture providing host-based services to client-based users. The architecture is based on the Software as a Service (SaaS) model where application software hosted on third-party servers is provided to users across the Internet. Both of these concepts loosely align with the Service Oriented Architecture (SOA) framework where functionality is organized by business processes that are discoverable via an independent registry. These schemes are essentially a reversal of the industry trend to push computing power and intelligence to the client.

Computing power has evolved and devolved over the past 50 years. In the 1950s the mainframe provided centralized computing accessible by “dumb” terminals. The 1960s saw the adoption of the minicomputer and the microcomputer. The minicomputer filled the niche between the mainframe and the microcomputer, often acting as a frontend to the mainframe, but users still used “dumb” terminals. The 1970s ushered in the age of the microcomputer whose progression with graphics empowered the user to work offline, remotely, and eventually in today’s ubiquitous mobile environments. In the 1980s a typical office environment was a desktop microcomputer that provided office applications, connected to the company’s local area network whose servers provided Internet access, and a “dumb” terminal emulator providing access to the mainframe for backend office applications. In the 1990s the World Wide Web opened up the Internet to third-party computing and set the stage for mobile computing. Browsers provided a “thin” client for interpreting a myriad of Standardized Graphic Markup Language (SGML) based languages developed by Internet developers. Client computing power grew from almost nothing in the 1950s to its peak in the 1980s and is now experiencing a decline as more applications migrate from the client to the cloud.

The 2000s have experienced increasing problems with maintaining software and ensuring a secure computing platform. Viruses, worms, spam, bug-laden software, and multi-vendor computing systems all contribute to the chaos. Client systems

are so complex today that bug fixes and security patches are almost a daily occurrence to upgrade hardware drivers, operating system components, communication interfaces, and application software. Vendors typically must maintain 7x24 online help desks to stay abreast of the constantly changing IT environments and their customers’ needs. As more and more companies offer online services via the Internet, fraud and IT vulnerabilities currently at an all time high will continue to increase. Arguments for cloud computing include the need to minimize client software maintenance and the need for a lower capability client such as a mobile phone that can offer the same application rich environment as a higher end client system such as a laptop.

Security attributes

The dependability of any application, especially one located in the cloud, is its ability to safeguard the users’ data. If the confidentiality, integrity, authenticity, and timeliness of the data can be validated, then the user can trust the cloud application. Further if the integrity, authenticity, and timeliness of the data can be proven to an independent third-party, we then have the ultimate security state of non-repudiation. From a cloud computing perspective, the securities attributes for preserving data, whether in motion or at rest, are interpreted as follows:

- **Confidentiality** – protecting data from unauthorized disclosure
- **Integrity** – ensuring detection of unauthorized modification
- **Authenticity** – validating the source (or destination) of data
- **Reliability** – verifying the integrity of data traceable to a valid point in time
- **Non-repudiation** – substantiating the authenticity and reliability of the data to an independent third-party

Confidentiality

Confidentiality for data at rest is arguably achievable via access controls that permit only authorized entities to view the data file. However, any data store is susceptible to unauthorized access if the access controls can be circumvented or disabled. An encrypted file cannot be viewed unless the adversary has access to the data encryption key (DEK); thus, the management of the DEK is critical. At the same time, relying on the same access controls for the DEK that were considered to be inadequate for the unencrypted file is not a reasonable solution. Since a cloud server by its very concept interacts with multiple clients, the DEK should not be the same for all clients as the compromise of a single DEK consequentially compromises all encrypted files of all the clients. A unique DEK per client would be a better scheme. The DEK granularity must be chosen based on security and operational parameters as a unique DEK per file provides a higher degree of security at the cost of more complex operations.

Confidentiality for data in motion is only achievable via encryption. The sender and receiver need to establish a transport encryption key (TEK) that is used by the sender to encrypt the file before transmission and by the receiver to decrypt the file when transmission is complete. When the file originates at the client as the sender, the client can establish a TEK with the server using a variety of key-management schemes. The client must be assured that the TEK has been established with the appropriate server and not another untrustworthy third-party. Likewise, when the file is downloadable to a client as a receiver, the server can establish a TEK with the client using a variety of key-management schemes. The server must be assured that the TEK has been established with the appropriate client and not another untrustworthy rogue client.

Integrity

Integrity of data consists of detection mechanisms as prevention is not possible. Data in motion across a communications line can always be altered, and similarly data at rest can be modified if the access controls can be circumvented or disabled. Even write-once-read-many (WORM) drives are indirectly susceptible to modification if the data path between the hardware and software viewer can be compromised. Thus detection is the only feasible approach. Integrity check values (ICV) are cryptographically derived values that are generated when the data is received or stored and can be validated whenever the integrity needs to be validated.

Cryptographic keys used with ICV are usually symmetric keys such that the integrity check key (ICK) must be shared between the entity that generated the ICV and the one validating the ICV. For example, the user generates an ICV on his client for a data file and then uploads both elements to the cloud server; the server then re-computes and compares the ICV to validate the data file's integrity. Another example would be a user validating the integrity of a file previously loaded to a cloud server; the client would request that the cloud server re-calculate the ICV; the client then compares

the ICV to validate the data file's integrity. For both examples the client and the cloud server need to establish an ICK using a variety of key-management schemes; and both parties must be assured that the ICK has been established with the appropriate corresponding entity. Note that symmetric schemes cannot provide authenticity since both parties share the ICK and either party can generate the ICV.

Authenticity

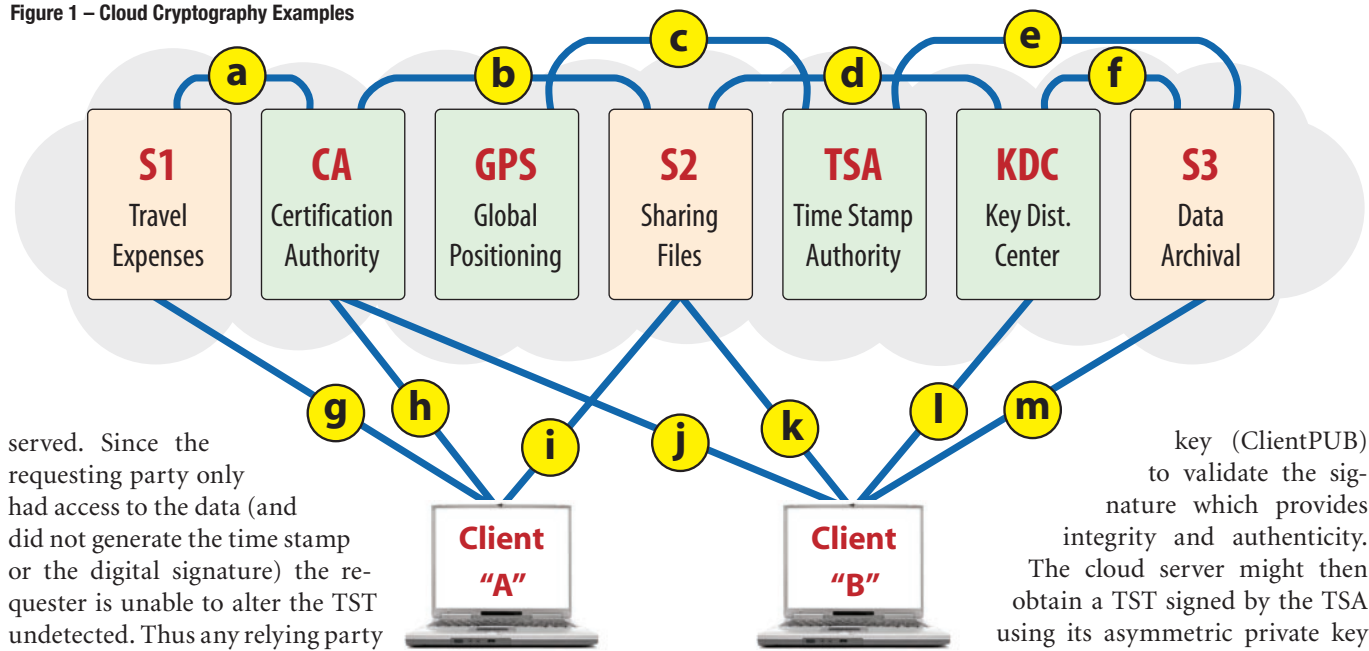
Authenticity between a client and a cloud server essentially boils down to the exchange of authentication data that is separate and possibly independent of the application data. For example, a user may log his client onto a cloud server via an identity (ID) and password, and once logged on may enter expenses via an online form; in this case the user's ID and password are authentication data separate and independent of the expense information which is application data. Another example would be a user who sends expense information to a cloud server that is digitally signed; for this scenario the digital signature is the separate authentication data but its creation is dependent on the application data. For both of these examples, the client has authenticated itself to the cloud server; however, this is insufficient as the server must likewise authenticate itself to the client.

Authenticity methods must also address the issue of continuity. Continuity must be preserved for each message exchange during the same interactive session and between sessions. The cloud server must be able to authenticate every message received from the client. Likewise the client must be able to authenticate every message received from the cloud server; but in addition the client must be able to dynamically authenticate any of its data stored on the cloud server. Authentication check values (ACV) are cryptographically derived values that are generated when the data is created or sent and can be validated whenever the authenticity needs to be validated. Digital signatures are often used for ACV schemes as the digital signature can only be generated by the holder of the asymmetric private key (ACVPRIV) whereas anyone with the corresponding asymmetric public key (ACVPUB) can validate the signature. Note that digital signatures can provide dual service as both the ICV and the ACV.

Reliability

Reliability is essentially a trusted time-stamp method incorporating a data integrity mechanism such that the data content is verifiable to a specific point in time. There are several trusted time-stamp methods available, all of which require the existence of an independent third entity known as a Time Stamp Authority (TSA). The TSA keeps a clock calibrated to the International Timing Authority (ITA); two of the ITA agents in the United States are NIST and the USNO which manages the GPS satellites. A time-stamp token (TST) consists of a TSA-generated time stamp cryptographically bound to a hash of the data, such as a digital signature generated by the TSA. Since the TSA only had access to the hash (and not the data) the TSA is unable to alter the data; its integrity is pre-

Figure 1 – Cloud Cryptography Examples



served. Since the requesting party only had access to the data (and did not generate the time stamp or the digital signature) the requester is unable to alter the TST undetected. Thus any relying party can verify that the data is unaltered and belongs to that specific TSA because the hash validates; and the relying party can verify when the data was established because the TSA digital signature validates.

Data reliability is a relatively new concept that has not yet been fully grasped or appreciated by the commercial or government industries. Suffice it to say that integrity without validation to a point in time is only valid for a very short interval to only a single entity. For example, an entity could receive a message, validate its integrity from the sender, and then alter the message for long-term storage. Unless another party had an authenticated copy of the original message and discovered the discrepancy, the stored version of the data might inadvertently be accepted as authentic. Another example might be an entity that generated two versions of the same document and digitally signs both versions. Since both digital signatures would verify, there would not be any reasonable way to discern which document was intended to be the original (or the correct version). For trusted time stamps to work properly in a cloud environment the TSA must establish its asymmetric public keys (TSAPUB) with cloud servers and clients so that its TST can be validated.

Non-repudiation

Non-repudiation is an older concept that has been around since the early days of the Public Key Infrastructure (PKI). There have been numerous articles and papers written on the subject but suffice it to say that the underlying concept is that the party who performed an action cannot deny the action. This means that the authenticity and the reliability of the action must be substantiated to an independent third party. It is often the case that the third party is presumed to be a judge in a court of law. Regardless, a user might generate a digital signature on its data using the client asymmetric private key (ClientPRIV) and upload the signed data to the cloud server. The cloud server might then use the client asymmetric public

key (ClientPUB) to validate the signature which provides integrity and authenticity. The cloud server might then obtain a TST signed by the TSA using its asymmetric private key (TSAPRIV) of the client digital signature which assures the reliability of the data and when the client signature was validated by the cloud server. Any party having the TSA asymmetric public key (TSAPUB) and the client asymmetric public key (ClientPUB) can substantiate the data's non-repudiation.

Cryptography services

Cloud application services must include the security attributes of confidentiality, integrity, authenticity, reliability, and non-repudiation and therefore must support cryptography solutions. At the same time there will be stand-alone cryptographic services in the cloud necessary to support application services and clients. Some of these cryptographic services already exist while others are still emerging. Figure 1 provides three examples of Cryptography as a Service (CaaS) floating in the cloud.

In Figure 1 there are three application services: S1, S2 and S3; four cryptography services: CA, GPS, TSA and KDC; and two clients: A and B. The interaction among the services and clients will be rather complex as demonstrated by the 13 interfaces labeled (a) through (m). Note that the two clients do not use all of the services and that the application services use some of the cryptographic services.

The application services and cryptographic service are described briefly as follows:

CA	Certification authority for generating and distributing public key certificates
S1	Application service for submitting travel expenses
GPS	Global positioning system – a constellation of satellites that provides a calibrated time source operated by the United States Naval Observatory
S2	Application service for sharing files

TSA	Time stamp authority for issuing time stamp tokens (TST)
S3	Application service for long-term data archival
KDC	Key distribution center for generating and distributing symmetric keys

The example interactions between clients, the application services, and the cryptographic services are described below. Note that numerous other possible scenarios exist.

Travel expenses

Client A wishes to use application service (S1) for submitting travel expenses; the following actions might take place to enable this business process:

- (a) Application service (S1) registers its public keys with the CA
- (h) Client A likewise registers its public keys with the CA
- (g) Client A and application service (S1) exchange certificates to establish a TEK for confidentiality and ACV for authenticity

Client A can then submit travel expenses to application service (S1) online with a high degree of assurance that the data is protected and authenticated.

Content distribution

Client A wishes to distribute content using application service (S2) as a short-term repository; the following actions might occur to enable this business process:

- (b) Application service (S2) registers its public keys with the CA.
- (d) Application service (S2) registers a one-time DEK with the KDC.
- (h) Client A registers its public keys with the CA.
- (i) Client A and application service (S2) exchange certificates to establish a TEK for confidentiality and Client (A) uploads content to application service (S2)
- (j) Client B registers its public keys with the CA
- (k) Client B and application service (S2) exchange certificates to establish an ACV and Client B downloads the encrypted content from application service (S2) along with a one-time password
- (l) Client B and KDC exchange certificates to establish a key encryption key (KEK) and an ACV; Client B submits the one-time password to the KDC to obtain the one-time DEK encrypted by the KEK

Client B can then recover the one-time DEK and decrypt the content. Any other client wishing to retrieve the same content would repeat the same steps as Client (B); however the application service (S2) would register a different one-time DEK with the KDC for the new client.

Data archiving

Client B wishes to archive content from its distribution source with application service (S2) to a long-term repository with application service (S3); the following actions might occur to enable this business process:

- (b) Application service (S2) registers its public keys with the CA
- (j) Client B registers its public keys with the CA
- (k) Client B and application service (S2) exchange certificates to establish an ACV and Client B downloads the encrypted content from application service (S2) along with a one-time password
- (m) Client B uploads the encrypted content to application service (S3) with instructions to obtain the one-time DEK from the KDC using the one-time password
- (f) Application (S3) using offline legacy processes (denoted by the dotted line) obtains the one-time DEK from the KDC using the one-time password and decrypts the content.
- (e) Application (S3) submits a hash of the content to the TSA to obtain a time stamp token (TST) and enters the content and TST into its database for long-term storage.

Client B now has its original content previously distributed by application service (S2) archived for long-term storage with application service (S3). Note that in this scenario the TST indicates the date and time of archival and not the previous content distribution.

Conclusion

The concept of Software as a Service (SaaS) has been extended to include Cryptography as a Service (CaaS) to address security for cloud computing. It is argued that such services are not only necessary but in fact already exist (i.e., CA, GPS, TSA). As more and more application services become available in the cloud, the need for cryptography-based security will continue to increase. At the same time, cryptography services will most likely specialize and proliferate, creating competition. Eventually as seems to be true with most segmented industries, some cryptography services will flourish, some will not, and others will be acquired to create mega-cryptography services.

About the Author

Jeff Stapleton is the CTO for Cryptographic Assurance Services LLC with over 25 years experience in the cryptography, security, financial and health care industries. He has participated in developing ISO and X9 American National Standards for over 20 years, the 10-year chair of the X9F4 working group, president and founder of the Information Assurance Consortium, and an ISSA member. He may be reached at jj78023@yahoo.com.

