International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

The science of data encryption – all terminologies

Yogesh Gautam
Pr. Scientist
ICAR-Directorate of Mushroom Research
Chambaghat, Solan Himachal Pradesh, INDIA
Yogesh.Gautam@icar.gov.in

Introduction

Terminology in the data protection world is often surprisingly imprecise. "Encryption" is something we all understand, or at least think we do: "A reversible algorithm that uses a key to make data unusable."Terms such as "tokenization" and "masking" and "obfuscation" are also used. Sometimes these refer to specific technologies; sometimes they are used generically, to mean "some form of data protection." For example, people often say things such as, tokenize/ encrypt/ mask/ obfuscate/protect the data"—without specifying how, and meaning different things at different times.

Even when the terms are used precisely, people often misunderstand the differences between them. And those differences matterIn this paper, all these terminologies are explained and also the relation amongst them

Obfuscate and protect

These are generic terms, meaning "The data is hidden somehow." That might mean replacing it with other data through encryption or another method, or simply not displaying it, perhaps showing just ***-**-**** for a US Social Security number.

Actually, when you're not intending to call out a specific technology, these are probably the best terms precisely because of their, well, imprecision.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

Mask

"Masking" is perhaps the most abused term. Sometimes people use it to mean "we keep the real data hidden" when they are actually just "obfuscating" or "protecting."

But masking can be more specific, referring to dynamic data masking (DDM) or static data masking (SDM). These technologies, conceptually similar, mean changing sensitive information in production data to other, similar values, at specific points in the data flow.

DDM occurs as the data is used:

- When a data feed is sent from one application to another, less trusted environment
- When data is displayed—for example, showing only the last four digits of a credit card number on screen in a call center application, so it can be used for account verification

While DDM has its place and is obviously better than nothing, using it as a primary means of data protection tends to give security, since it means that a database breach is guaranteed to reveal cleartext.

SDM means copying production data (in toto or a subset) and replacing sensitive information with other values. Typically this is done to enable realistic test conditions based on production volume, variability, etc., without risking exposure.

Some SDM tools preserve referential integrity; that is, all occurrences of "Bob" (and only "Bob") might be changed to "Thomas." Others change values without consistency. Sometimes "Bob" becomes "Thomas," and other times it becomes "Frank"; perhaps both "Bob" and "Thomas" become "Frank," consistently or otherwise.

In either case, the resulting data can be used without regard for security, since it no longer contains actual sensitive values.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

Encrypt

People are generally conceptually familiar with encryption, which combines a mathematical algorithm and some other, secret information (an encryption key) to transform data in a way that is essentially impossible to reverse without the appropriate key.

Traditionally, this meant that the data was converted to an unreadable binary string, usually longer than the input. If a whole files system, database, or dataset was encrypted, and then decrypted, either manually or automatically (transparently), on use, this change usually had no significant impact.

On the other hand, when adding field-level encryption to existing applications and data stores, this obviously had significant impact: Program variables, database column definitions, and file layouts all must be changed, and disk space usage increases, sometimes quite dramatically.

Format-preserving encryption, pioneered by Voltage Security in the late 2000s, solves this last problem, producing ciphertext that maintains the input size and alphabet.

Tokenize

"Tokenization" as a specific term comes from the Payment Card Industry Data Security Standard (PCI DSS), a well-established standard for protecting payments-related data.

When PCI DSS v1.0 was released in 2008, it required merchants to "render [credit card Primary Account Numbers, or PANs], at minimum, unreadable anywhere it is stored … [using] strong one-way hash functions (hashed indexes), truncation, index tokens and pads … [or] strong cryptography."

These choices obviously represent different use cases: Hashes are great for anonymizing data, but their irreversibility makes them somewhat less than useful if the cleartext will ever be needed again.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

Truncation means storing only part of the data. Again, this only works if the full cleartext is never required.

Index tokens and pads are what people usually call tokenization: replacing the value with a randomly generated value. More current versions of PCI DSS describe this approach in more detail:

An index token is a cryptographic token that replaces the PAN based on a given index for an unpredictable value. A one-time pad is a system in which a randomly generated private key is used only once to encrypt a message that is then decrypted using a matching one-time pad and key.

How tokens and encryption relate

Index tokens are a one-time pad: that's the "unpredictable value." Since PCI DSS is focused on credit card numbers, this "index" can be thought of as an "offset": Take a given card number, add some value to it ("wrapping around" if it exceeds the maximum possible), and there's your token.

Typically this is implemented using a database; a random number generator creates a value that's in the desired range representing the PAN, and that value is stored in the database.

What is critical to realize is that in this scenario, the database is an encryption key. Just because it is not a 128-bit, 256-bit, or some other defined size does not mean it is not an encryption key.

Given an algorithm and some secret information (in this case, the database), a cleartext PAN can be consistently converted to a token and vice versa. That is precisely the definition of encryption.

Encryption vs. tokenization: Making the choice

If tokenization is encryption, why do people distinguish between them? PCI DSS auditors (qualified security assessors or QSAs) often prefer tokenization over encryption, citing an alleged difference in that there is "no mathematical relationship" between a randomly generated token and its cleartext.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

This is flawed reasoning. That relationship exists, in the form of the one-time pad represented by the tokenization database or metadata table.

In addition, the first format-preserving data protection methods available in the early days of PCI DSS were database-backed tokenization schemes. This first-mover advantage has probably further encouraged QSAs to prefer tokenization.

Local operation vs. remote

But there is still a big difference between most tokenization and encryption solutions—whether the operations are performed on the endpoint or not.

This matters because of accessibility of the key material (whether an AES key or a tokenization table of any sort). With local operation, that key material must be present on the endpoint. If operations are performed remotely (typically via web services such as REST or SOAP), that key material is more secure, since it remains on the remote server.

Of course this does not mean remote operation is invulnerable; a flawed implementation could allow attackers to issue requests to convert protected data to cleartext, and of course they can attack the server directly.

But a large number of remote requests from an intruder will typically be visible on several levels (network traffic, server logs, etc.). In addition, such servers are usually more tightly controlled than endpoints, typically being administered by network or security staff, with quick attention given to known security vulnerabilities. They are also generally single-purpose, further reducing the odds of compromise.

Encryption is frequently offered both locally and as a web service. In the former case, the key material is downloaded from a server and used for local operations. Local operation is robust and fast, since it eliminates network latency once that key is initially downloaded.

On the other hand, most tokenization is offered only via web services. For database-backed tokenization, the reason is obvious: Multiple endpoints need to tokenize and detokenize, so they need a single point of control that owns the token database. That database also needs care and

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

feeding, as well as (hopefully) some sort of real-time replication, to avoid lost tokens should a hardware failure occur.

Understanding scalability issues

Database tokenization also suffers from scaling problems. The tokenization sequence generally flows:

1. Cleartext value received.
2. Database checked for existing token for that value; if found, token returned.
3. If not found, generate new token, search database to make sure not a collision (same token representing two cleartext values).
4. If collision found, repeat previous step.

As the number of issued tokens increases, collisions occur more frequently. When tokenizing values such as PANs, where the bulk of the possible values exist, this becomes a serious problem.

Dealing with synchronization problems

Worse, however, are synchronization issues. For scalability as well as reliability, tokenization servers must be replicated, introducing two new problems. First, the same cleartext PAN could arrive at two different servers simultaneously, resulting in two different tokens being generated.

While this would likely eventually be resolved, the inverse is far more pernicious: Two different cleartext PANs can hit two servers and produce the same token. These issues can be avoided through real-time database replication, but that adds further complexity and latency.

Non-database-backed tokenization approaches such as Voltage secure stateless tokenization (SST) allow both remote and local operation. Given a copy of the tokenization metadata, an endpoint can perform tokenization while guaranteeing consistency with other machines and avoiding real-time replication requirements.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

This approach also avoids the scaling problems endemic to database-backed systems, including extended token generation times as the number of tokens grows and the expense and delays of real-time database replication (or, worse, consistency issues if such replication is not implemented).

Local machines more at risk

Since tokenization is encryption, local tokenization and local encryption are identical from a security standpoint. In both cases, however, there is sufficient information on the local machine to allow serious data theft should that machine be compromised.

Thus, any local data protection on an endpoint must ensure that this information—encryption keys and/or tokenization metadata—is sufficiently protected that it not become a primary attack surface. Ironically, because tokenization metadata is larger than an AES key, it "feels" like it's more vulnerable. Yet because they are so much smaller, exfiltration of AES keys would be far easier!

The reason QSAs dislike endpoint data protection should be clear: Since most such machines are not as carefully controlled as single-purpose servers, there is higher risk of compromise, either through local, seemingly unrelated configuration changes, or because some other application or service running on that endpoint is vulnerable.

When considering operation in the cloud, this becomes even more significant, as vulnerabilities like SPECTRE/MELTDOWN are impossible for an administrator to protect against. Attacks can come from unknown and unrelated code that happens to be running on the same hardware.

Many scenarios make local operation desirable, since it offers higher performance and more predictable response times, avoiding network latency/congestion issues. PCI DSS QSAs are less likely to sign off on local protection, but other use cases can benefit. This may require hardening the endpoint, perhaps even requiring removal of other services, to satisfy regulatory security requirements.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

Understanding the differences is key

Data protection terminology can be precise, and communication is clearest when it is used this way. Understanding the differences among technologies is critical.

For regulatory compliance, it is important to understand that the most important difference is often between local and remote operation, rather than between encryption and tokenization.

Data Masking

In 2017, a breach of Equifax through an unpatched server led to the exposure of sensitive data on more than 143 million US citizens, representing more than half of all adult Americans and exposing information such as Social Security numbers.

The breach led to a tepid $700 million fine, much debate over the future of SSNs as a de facto national identifier, and companies re-evaluating how to protect information in a usable way. One solution, data masking—a technique for hiding sensitive data from being viewed by unauthorized users—has taken off and evolved since then. Data masking has morphed from simple replacement of sensitive data to dynamic masking and tokenization techniques that allow companies to preserve much of the usefulness of data while protecting information from attackers.

When done correctly and paired with the right data-security policies, data masking can allow protected data to still be functional, said GouthamBelliappa, vice president of AI engineering for Capgemini Americas, an IT services firm.

The move to data masking is important because it gives companies a choice. In the past, companies would have to either delete data and lose its value, encrypt data and make its value harder to extract, or attempt to protect data and be at risk of a data breach.

Because encryption has historically been hard to manage, the choice resulted in companies often rolling the dice, said Reiner Kappenberger, director of data-security products for Micro Focus.

Data masking can help solve problems with data retention, but companies need to educate themselves. Here are four key issues which need to be understood.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

1. Data masking covers a lot of ground

The technologies covered by the term *data masking* have expanded. Originally, data masking often just referred to hiding sensitive data from prying eyes—sending "(000) 000-0000" instead of a phone number, for example. Increasingly, data masking also encompasses functional anonymization and pseudonymous capabilities, producing tokens for sensitive data values that cannot be reversed but allow for some analytical capabilities. Data masking makes use of techniques such as format-preserving encryption (FPE) and stateless tokenization.

These techniques are practical, unlike calculation-intensive techniques such as homomorphic encryption, which can cause both the size of data to grow by at least two orders of magnitude and the complexity of computation to explode.

2. Not all data masking is the same

Static data masking uses a second database with masked data that is used to deliver records to normal users, while privileged users still have access to the original data. Dynamic data masking programmatically changes the data when it is requested, using a variety of techniques that can preserve some of the usefulness of the data. Tokens may be used in place of credit-card numbers, for example, allowing one-time transactions and preventing the leakage of account information.

While such techniques provide strong capabilities, they are not universal, said Belliappa. Many database products offer data-masking features, but more often than not, the features essentially allow the literal "masking" of data to hide the information. Other dynamic data masking uses machine-learning algorithms to change the data as it is delivered, but poor training can affect the eventual result.

3. Good data masking should satisfy current regulations

The eventual goal of data masking is to prevent the need for notification when a breach happens. In its 2021 Guidelines on Examples Regarding Data Breach Notification, the European Data Protection Board confirmed that a company that loses or leaks encrypted data but still retains access to the data does not need to notify customers of a breach.

International Journal of Research in IT and Management (IJRIM)
Available online at: http://euroasiapub.org
Vol. 11 Issue 10, October- 2021
ISSN(o): 2231-4334 | ISSN(p): 2349-6517 | Impact Factor: 8.106

To take advantage of such legal protections, however, companies should make sure that they are taking the proper steps and using the right technologies.

4. It needs to be made sure that data can still be used

Finally, companies need to ensure that masked data will still serve the needs of its users. Employees who are not satisfied with data-masking technology may try to do an end run around the technology. The masked data needs to give the same results as real-world data and not be biased or skewed as a result of the masking, said Capgemini's Belliappa.

Take a stand

Companies should start taking a harder stance on their data—if they don't have a use for it, then delete it. Yet, in all other cases, masking the data will help companies escape the worst of a breach.