Some Privacy Practices May Result in Under-Reporting of Breach Incidents

By Kelce S. Wilson, CIPP/E, CIPP/US, CIPM

There is a belief, at least among some privacy practitioners, that, if data was encrypted when it was stolen by a hacker, then the theft incident does not meet the legal definition of a data breach and therefore does not trigger reporting and notification obligations. The reasoning is that encryption preserves confidentiality by rendering the data unreadable by hackers, and so there is no reasonable likelihood of harm to the data subjects. Privacy compliance practices that incorporate this belief are bolstered by an "encryption exception" to reporting requirements in some regulations, such as Article 34(3)(a) of the EU General Data Protection Regulation and some U.S. state laws.

Unfortunately, though, the reasoning is quite often mistaken, as explained in a chart provided below that identifies seven different vulnerability scenarios whereby encryption does not preserve confidentiality of stolen data. Simply put, many widespread key management practices are not sufficiently secure to justify assuming that the stolen data cannot be decrypted and thus fully exposed and exploited. If the legal or compliance expert, who is handling the incident response, does not thoroughly investigate the encryption system and key management practices that were implemented, there is no way to properly ascertain whether a potentially applicable encryption exception is being properly applied or misused to avoid reporting a legitimate breach.

The exploitability of a weak key management practice is something that I know first-hand. At an earlier point in my career, when I was working as a white hat hacker (aka "red team penetration tester") and test planner, I was presented with a challenge: Try to crack a secure launcher (an anti-hacking protection measure), which used AES-256 encryption to protect a critical file. Since AES-256 is widely considered to be a highly secure encryption method that requires billions of years to crack in a brute force attack, the people who presented the challenge to me expressed confidence that I would be unsuccessful.

However, I beat the encryption in only seven minutes, totally defeating the protection. How could I do that? There was a human error in the key management process. The intended key management system was solid … on paper, anyway. But humans were involved, and it required only just one of them to make a single, easy mistake to enable me to compromise the key.

Included within the description of the vulnerability chart below is a high-level description of the hacking tool that I created to exploit the key management error. I seriously doubt that I was the first one to have had such an idea; instead, I expect many malicious hackers have already created and used similar tools. But fear not, because now, anyone who is even minimally competent in hacking and programming, and who reads this article, will be able to easily create their own version of the tool to use against your encrypted data.

Nice thought, isn't it? So … perhaps you might want to reconsider your reliance on the encryption exception or at least contemplate the suggestions that follow the chart.

In defense of privacy practitioners who had previously adhered to the belief that encryption renders data theft merely an incident rather than a breach, several laws and regulations tout encryption as a data security measure without thoroughly emphasizing how high the risk of key compromise can be in many scenarios.

Several laws and regulations tout encryption as a data security measure without thoroughly emphasizing how high the risk of key compromise can be in many scenarios

For example, the April 27, 2016, version of the EU General Data Protection Regulation itself does not mention key management considerations when describing its version of the encryption exception, although it is addressed by the Article 29 Working Party. Article 34(1) of the GDPR identifies a notification trigger:

> **Communication of a personal data breach to the data subject**
>
> 1. When the personal **data breach is likely to result in a high risk** to the rights and freedoms of natural persons, the controller shall communicate the personal data breach to the data subject without undue delay. (emphasis added)

And then, just a paragraph later, Article 34(3)(a) states the exception for encrypted data:

> 3. The **communication** to the data subject referred to in paragraph 1 **shall not be required if** any of the following conditions are met: (a) **the controller has implemented** appropriate technical and organisational protection measures … in particular those that render the personal data unintelligible to any person who is not authorised to access it, such as **encryption**; …. (emphasis added)

This description of the exception does not explicitly mention requiring sufficiently secure key management practices that theft or other compromise of the key is rendered unlikely. Other parts of the GDPR also mention encryption, such as the Whereas clause 83, Article 6(4), and Article 32(1). However, none of those passages mentions secure key management practices either. The Article 29 Working Party guidelines did address key confidentiality October 3, 2017, requiring that "the key was not compromised in any security breach…."

If looking only at the text of the GDPR, though, a company can experience a theft of encrypted data that truly should be defined as a breach because the decryption key has likely also been compromised, but yet the company might consider itself exempted from any reporting or notification obligation. This situation then becomes an unreported breach, contributing to potential under-reporting. Given the severity of potential GDPR penalties, how many data protection officers might be willing to eagerly embrace the exception without thoroughly investigating all potential vulnerability scenarios?

How many DPOs even know to consider this type of risk? How thoroughly is key management addressed in your risk assessments? This creates an issue that compliance does not necessarily mean security, when compliance personnel merely works with checklists without having a proper technical comprehension of real-world hacking threats.

The problem of potential breach under-reporting can also reach into the U.S., although the 2012 version of the book "U.S. Private-Sector Privacy," published by the International Association of Privacy Professionals, does properly identify that the key must remain secure. Specifically, page 83 of the book states:

> **Most states exempt** individuals and businesses from data breach notification and disclosure requirements **if the data was encrypted** when lost, …. However, the encryption exception typically applies **only when the key remains secure**. Most states make this explicit by stating that **the exception does not apply when the decryption key is breached** along with the encrypted data. (emphasis added)

Although this guidance does reflect a more thorough comprehension of the threat situation, it does not address all of the likely vulnerabilities, nor does it highlight that the key is likely vulnerable to theft or compromise in many — if not most — common data processing scenarios. The vulnerability chart identifies four encryption scenarios (A–D) and seven vulnerabilities (1–7), and indicates degrees of vulnerability for each pairing.

The left-most two encryption scenarios, A and B, in which encrypted data is accessible by a processor and a software application that operates on that data in cleartext form (unencrypted or decrypted state), is the most common. This is because encryption scenarios A and B are the common situations in which data is accessible for use in processing. It can be viewed, edited and otherwise used. The right-most two encryption scenarios, C and D, are typically only encountered in offline backup archive situations.

For example, if a database is stored on a network drive, encrypted at rest, but is automatically decrypted when someone attempts to access the database, this is encryption scenario A. If a password is needed to open the file, this is encryption scenario B. The physical action in that scenario is the typing of the password, which is then used to generate the decryption key in a "just-in-time" manner. A less common but still somewhat widely used version of encryption scenario B is that the decryption key is stored on a piece of hardware, such as a USB dongle or optical disk. Perhaps the most common version of encryption scenario B, though, is a notebook computer with an encrypted the hard drive for which a password is required to unlock the hard drive encryption.

Note that there are differences between moving a key "out of band" so that it is not co-located with the encrypted data. Encryption scenarios A and B are identified in the chart as separate, but there are actually myriad scenarios that are a blend, perhaps closer to one of the scenarios than the other.

Some organizations create backups of their data and store those back-ups in an archive that is both off-site and offline. If the backup is encrypted and the key is never placed on any node that is accessible to the network on which the backup is stored, this is encryption scenario D. This scenario

	Encryption Scenario			
	Encrypted data is accessible by a processor and software application that operates on it in cleartext form.		Encrypted data is not accessible by a processor or software application that operates on it in cleartext form.	
Method of Compromise	A. Key material is accessible by a processor and automatically decrypts data for use by software. **Common**	B. Key material is not accessible by a processor until physical action decrypts data for use by software. **Common**	C. Key material is accessible by the same network on which the data is archived.	D. Key material is not accessible by same network on which data is archived, but is kept physically separate.
1. Data and key are both stolen at the same time.	**Vulnerable; easy to decrypt**	**Prevented**	**Vulnerable; easy to decrypt**	**Prevented**
2. Data and key are stolen in same compromise, different times.	**Vulnerable to a persistent threat**	**Vulnerable to a persistent threat**	**Vulnerable to a persistent threat**	**Prevented**
3. Data is stolen when it is used by software in cleartext form.	**Vulnerable; timing is important**	**Vulnerable; timing is important**	**Prevented**	**Prevented**
4. Data and key are stolen separately in different incidents.	**Vulnerable; possibly delayed compromise**	**Vulnerable; possibly delayed compromise**	**Vulnerable; possibly delayed compromise**	**Vulnerable; possibly delayed compromise**
5. Encryption is weak; less than 256 bit.	**Vulnerable; easy to decrypt**	**Vulnerable; easy to decrypt**	**Vulnerable; easy to decrypt**	**Vulnerable; easy to decrypt**
6. Password is weak; under 15 characters.	**Vulnerable; easy to decrypt**	**Vulnerable; easy to decrypt**	**Vulnerable; easy to decrypt**	**Vulnerable; easy to decrypt**
7. Sophisticated attack limits key entropy, permits guessing of key.	**Vulnerable; advanced stealth threat**	**Vulnerable; advanced stealth threat**	**Vulnerable; advanced stealth threat**	**Vulnerable; advanced stealth threat**

is the one with the lowest likelihood of key compromise. Encryption scenario C is included for conceptual completeness of the chart, although it indicates a poor security mindset and should hopefully be rare: The backup is stored offline but the decryption key is stored along with the encrypted data.

A glance at the chart reveals that the common encryption scenarios A and B are the most vulnerable to compromise of the decryption key.

To summarize, the lesson here is that if the encrypted data is stolen from an off-site, offline backup archive, and the key was not accessible by anyone who compromised that system, then the reasoning behind the encryption exception is more likely to be valid than if the data was stolen from a working directory of a system on which the data was accessible for processing.

More analysis is needed, though, because among other risks, there is a possibility that the decryption key could have been obtained via a different incident, even for the most secure of the encryption scenarios, scenario D.

We now move though explanations of the vulnerabilities.

As a side note, the chart uses the phrase "key material" to mean both a copy of the key itself or, for systems that generate keys for use and then erases them from memory, the secret material that is used for generating the key whenever it is needed. A password that is typed into a keyboard by a human is an example of key material that is used (often via a hash algorithm) to generate a key. There are other, more complex versions of key material use, but the concept is that either the key itself or the key material contains the secret information that is needed for decryption.

Another note of explanation is that, for some encrypted data, the encryption key is the same as the decryption key. This is symmetric encryption, but there are also systems in which the encryption and decryption keys are different.

Vulnerability #1: Data and key are both stolen at the same time.

The simple matter is that if authorized users can automatically decrypt the data to read, edit or otherwise use the data, then the key must necessarily be accessible by at least one system on which the data is stored and the software executes. Even a novice hacker will have a chance of locating and stealing the key.

In the seven-minute defeat mentioned earlier, the people issuing me the challenge had planned to create a version of encryption scenario B, in which the decryption key was kept only on an optical disk. The challenge was for me to open the file (to run an encrypted executable program) without the benefit of the decryption key on the optical disk. However, AES-256 is symmetric encryption, so the decryption key was the same as the encryption key. Someone who encrypted the file forgot to delete the encryption key, thereby inadvertently creating what the chart describes as encryption scenario A.

This was nothing more than a simple human mistake. And which of us has never forgotten to do something? It was such an easy mistake — and likely one that could also occur with your own systems, unfortunately.

Encryption and decryption keys have certain properties that make them easily distinguishable from other types of computer files. For example, if you are

reading this on a computing device, it may have hundreds of thousands of files on it. Common files, such as word processing documents, JPEG images and audio files, typically have a defined file format. Executable program files typically contain a high percentage of hexadecimal values that correspond to machine language instructions and, due to the operation of compilers, typically contain a large number of no-operation instructions. For example, programs designed to run on Intel processors have a high occurrence of the hexadecimal number 90, which is the machine language represented by the assembly language mnemonic NOP.

In anticipation of the challenge, I had written a program that would search through all the files accessible on whatever system I would target, reject those having a defined-file format or a high occurrence of a limited set of hexadecimal values, and thus rapidly identify files having high entropy (i.e., a high degree of randomness). When I started the challenge, it was easy to identify which one contained the key that had been left by human error.

As a side note, if a password manager keeps a local copy of the key library (although in an encrypted state — that is the keys are encrypted by a secondary encryption), and multiple keys are in the key library, then the file size will be multiple times larger than the length of the key being sought by a hacker. However, it will likely still contain high entropy, meaning that it will have a high degree of randomness.

If hackers can get into the system on which the data is processed, and users who process that data are able to decrypt it in order to work with it, then the key is accessible. If the hackers grab everything, they might easily take the decryption key material at the same time and can search through their own copies of the stolen files, at their leisure, using a tool that is designed to identify possible key material. Therefore, it is important that, during an incident breach response, the investigation does not stop at looking over the key management policy but continues on to perform forensics of the actual key management activities and searches for a copy on all systems that the intruder could have accessed.

Common encryption scenario A is easily vulnerable to this type of threat.

Vulnerability #2: Data and key are stolen in same compromise but at different times.

In the situation of a persistent threat, in which an intruder keeps a presence on a compromised system for an extended period of time, then encryption scenario B also becomes vulnerable. In some situations, a hacker might only be in the system for a short period of time before leaving or being discovered and blocked.

If hackers can get into the system on which the data is processed, and users who process that data are able to decrypt it in order to work with it, then the key is accessible

However, persistent threats, such as the 10-year presence of Chinese spyware on Nortel's computer systems, are quite common. So, even if the decryption key is not automatically accessible whenever a user attempts to open an encrypted data file, the key can easily be compromised if a user decrypts the data even a single time during the period in which a hacker has access.

Vulnerability #3: Data is stolen when it is used by software in cleartext form.

In some situations, a skilled hacker can grab data when it is in decrypted form without ever requiring the decryption key. Here is one example of how that can work:

You believe your email is relatively secure because it is encrypted (both incoming and outgoing), and local copies are stored only in encrypted form. The key is carefully managed by a program having sufficient security to defeat most hackers' best efforts. However, if your email system permits multiple simultaneous login sessions and setting rules for incoming and outgoing messages (such as automatically routing messages to certain folders).

If a hacker ascertains your login credentials somehow, logs in to your email account, and sets two rules — one to send all incoming mail to your trash folder and another to turn off notification of unread emails that are in the trash folder — then the hacker monitors the trash folder, while you are looking only at the inbox and are entirely unaware of the problem. As a new email comes in, it is routed to the trash folder and decrypted for the hacker to read.

The hacker can use the email program to copy the data and files to their own system and then move the new message into the inbox, where you see it and mistakenly believe it to have just arrived. In this situation, the hacker has defeated both the encryption of the email during transmission, as well as the local email storage, all without ever needing to compromise any decryption key.

Did you have any idea that a data compromise could be so easy? So then … what are your thoughts about that encryption exception now? And we're not done with the bad news, yet.

Vulnerability #4: Data and key are stolen separately in different incidents.

Here is a threat situation to which all of the encryption scenarios — A through D — are vulnerable: The hacker gets the data at one time, and in a separate (possibly undetected!) incident, obtains the key separately. There are some important things to contemplate:

1. Incidents should not be examined alone but rather should be examined for possible data and key pairings being stolen in different incidents.

    a. If data had been stolen, check whether the decryption key for that data might have been stolen during a prior incident, and

    b. If a decryption key had been stolen, then all prior compromises of data that can now be decrypted, suddenly becomes a breach – even if you had properly qualified for the encryption exception during those earlier incidents.

2. If the data had been encrypted with symmetric encryption, it might be a good idea to stop using the key chat corresponds to the stolen data so that all copies can be destroyed. Not deleted but thoroughly destroyed. If you keep a "just-in-case" copy for yourself, then it might be compromised in a future incident.

Just a question here: Does your incident response plan include these suggestions of analyzing multiple incidents for separate data and key theft events that could, in combination, result in a compromise? What about changing keys, even if only data was stolen and the key had remained secure?

Vulnerability #5: Encryption is weak, less than 256 bit.

I have heard reports about companies that securely dispose of used hard drives from hospitals finding that the data is encrypted with 16-bit encryption. Think about that for a minute.

16-bit encryption is so trivially easy to crack with modern computers that it is effectively useless for anything other than annoying the hackers. It certainly won't stop the good ones.

But it's compliant, right? What does the GDPR say about using 16-bit encryption instead of 256? Can you find anything specifying minimum key length? The Article 29 Working Party guidelines clarified as "state-of-the-art encryption," "appropriate level of encryption," and "considered currently adequate by security experts." Presumably, this can be interpreted as 256-bit or higher with a vetted algorithm.

Vulnerability #6: Password is weak, under 15 characters.

How clever is your password? Does it use non-standard characters? Good. But if it is less than 15 characters, no matter how bizarre the set of characters, someone has already included it in a list that hackers use, called a rainbow table. Rainbow tables are lists of password data (precomputed hash values) that can be used for rapidly breaking into many systems that require password logins.

A good suggestion for passwords is to use one that is 15 characters long or more, along with a secure password manager. To make the password easier to remember, consider stringing together a set of four misspelled words that you can associate with an image in your mind. Something silly, painful or exaggerated in dimensions will be easier to remember. Try something like a description of a person, a profession, an action and some object.

16-bit encryption is so trivially easy to crack with modern computers that it is effectively useless for anything other than annoying the hackers

An example could be visualizing a barber using the scissors to cut off a person's ear lobes, instead of their hair, and then type "tall*barber*cutting*ears." This is a horrifying thought, but if you concentrate on a visualization for even just a few seconds, it will be easy to remember. Pick something that rings unique to you and is similarly easy to remember for your important passwords.

Vulnerability #7: Sophisticated attack limits key entropy, permits guessing of key.

This type of compromise is exceptionally stealthy and can persist for years without detection. To implement it, a sophisticated hacker inserts some type of program onto your computer system so that, whenever one of your programs generates an encryption key, the key is limited to being one of a small set of possibilities. To understand this type of hack, you need to understand the difference between key length and key entropy.

Have you ever seen one of the encryption keys you rely upon to keep your data secure? Probably not. And even if you did look at it, it would appear to be random gibberish to you. The security of encryption is provided by the key being able to take on so many different possible values that a hacker or eavesdropper cannot possibly guess all of them.

Computers process instructions and data in a deterministic manner rather than randomly. However, to generate a good encryption key, the computer needs to locate a source of random information and convert it into data for key material. One way is for the computer to use some type of interaction with a human, such as measuring the hundredths of a second between keyboard presses when the person is typing. Some computers have special circuitry that takes some measurement (such as temperature or electrical noise impulses) and convert these into random numbers.

The problem is that whatever process converts these sources of random information into an encryption key is often implemented in software. If a sophisticated hacker can modify that software so that whatever random data is supposedly used, the randomness it is limited down to is only a few thousand possibilities. Limiting the degree of randomness in some data stream is one way to limit entropy. While it may be computationally infeasible to guess all possible keys in a state-of-the-art encryption scheme, guessing only a few thousand is relatively easy. And unfortunately, a human looking at encryption keys cannot reliably ascertain whether the set of keys has been limited to a set of possibilities that is computationally feasible to crack in a brute force attack.

Imagine such a hack had been implemented on your computer system. How would you ever know? Even if you generated and compared thousands of encryption keys, as some sort of test, you might not detect it.

So, if a good hacker knew they would be eavesdropping on some company's email traffic or breaking in to their systems to steal encrypted data, they might invest the time in attempting to surreptitiously modify the source of the entropy used by whatever random number generator that the encryption program used for generating keys. Then, they could read all the stolen (or intercepted) data without anyone having any idea.

Interestingly, the Article 29 Data Protection Working Party guidelines from October 3, 2017, appear to address something similar to this type of vulnerability:

> However, **if the confidentiality of the key is intact** — i.e., the key was not compromised in any security breach, and was **generated so that it cannot be ascertained by available technical means** by any person who is not authorised to access it – then the data are in principle unintelligible. (emphasis added)

Conclusion

In view of all these vulnerabilities, it is apparent that some dedicated analysis may be required for deciding whether a theft of encrypted data is merely an incident, qualifying for the encryption exception, or might instead be a full-fledged breach. An overly simplistic reasoning that the data was encrypted, so there is no breach — without analyzing key management risks — may cause under-reporting of breach incidents.