Does the Cryptographic Hashing of Passwords Qualify for Statutory Breach Notification Safe Harbor?

Jason R. Wool*

Data breaches are increasingly part of the news cycle, especially as "mega breaches" occur from time to time impacting tens of millions of people (two recent examples include OPM and Equifax). Today, 48 states have passed legislation requiring notifications to consumers (and sometimes regulators) following a data breach, in addition to several non-state jurisdictions like D.C. and Puerto Rico. In light of the ubiquity of these laws, those unfamiliar with data breach-related legal practice may be forgiven for assuming there is a rich body of case law interpreting state notification statutes. Surprisingly, however, there are very few cases analyzing the nuts and bolts of these laws, leaving some quirks in the various statutes to be analyzed by lawyers in a guidance vacuum.

Password hashing is one such mystery of state data breach notification law analysis. Every state with a breach notification law effectively provides an exemption from statutory breach notification requirements when breached "personal information" is encrypted and the encryption key

* *Jason Wool, CIPP/US, CIPP/E,* is Counsel at ZwillGen PLLC. Jason's practice focuses on information security, including cyber risk management, incident response, and compliance with global data protection laws, regulations, and standards, including the PCI-DSS. He holds a B.A. from Haverford College and a J.D. from William & Mary. The views expressed in this article are solely those of the author, and do not necessarily represent those of any organization with which the author may be associated.

remains secure (referred to as "safe harbor"). However, none of the jurisdictions that consider some form of a username and password to be personal information (and therefore potentially subject to data breach notification obligations) explicitly refer to password hashing in their statutes, even though most organizations hash, rather than encrypt, passwords.[1] Personal information, as defined in state breach notification statutes, has traditionally consisted of data such as social security numbers, driver's license numbers, and financial account numbers. This data would typically be encrypted rather than hashed as a data protection measure. But as user credential breaches become increasingly prevalent, some states have begun bringing this type of information in the scope of mandatory breach notification.

This article analyzes two under-explored aspects of password hashing in the context of state breach notification laws. First, it explores whether password hashing can qualify for data breach notification "safe harbor" provisions, which typically apply where data is encrypted. This article concludes that password hashing can generally qualify for safe harbor, but that the argument is more difficult to make in three of the states examined. Second, even if password hashing can qualify for safe harbor under these laws, it explores whether hashing always qualifies or if it does only when certain minimum criteria are met. This article concludes that minimum criteria must reasonably be met, given that some hashes are trivial to crack. The state breach notification laws do not provide much clarity on this point.

Ultimately, this article identifies as a statutory

[1] At the time of writing, at least eight jurisdictions – California, Florida, Illinois, Nebraska, Nevada, Puerto Rico, Rhode Island, and Wyoming – consider some form of a user ID and password to be "personal information."

shortcoming the fact that the state breach notification laws in jurisdictions that consider user credentials to be personal information do not always clearly acknowledge the reality that hashing is used to protect the data in question. Moreover, they also do not provide sufficient criteria for assessing the effectiveness of hashing with respect to determining the applicability of safe harbor provisions. As bulk user credential data breaches become more common, the lack of statutory clarity around hashing may give rise to significant uncertainty about when regulators consider notification under the state breach notification laws to be required for incidents involving account credentials.

I. OVERVIEW OF BREACH NOTIFICATION LAWS AND SAFE HARBOR PROVISIONS

Breach notification laws typically require notification to individuals impacted by a data breach, as well as to regulators like state attorneys general in some cases. The laws typically define the set of data elements that can be subject to a data breach and usually call those data elements "personal information."[2] The statutes also typically define a data breach (or some similar term), often using language such as "unauthorized acquisition, or reasonable belief of unauthorized acquisition, of personal information that compromises the security, confidentiality, or integrity of the personal information maintained by the information collector."[3]

All breach notification statutes in the U.S. contain provisions clarifying that notification is not required if the data was protected in certain ways. These are sometimes

[2] *See, e.g.*, Alaska Stat. § 45.48.090(7) (2016).
[3] *Id.*

referred to as "safe harbor" provisions. There are a number of variations of the safe harbor provision in state breach notification laws, but they generally all apply to data that is encrypted regardless of the specific language used. In some cases safe harbor is provided explicitly where personal information that was subject to a data breach was encrypted and the encryption key was not compromised or stolen.[4] In some of these statutes, the term "encryption" is defined,[5] however, others do not define it.[6] Some statutes provide safe harbor but do not use the term encryption at all.[7] Other statutes provide safe harbor where personal information is encrypted, redacted, or otherwise rendered unreadable.[8] Regardless of the specific language used, encryption of personal data generally qualifies for safe harbor in the event of a data breach (assuming the key was not compromised or stolen).[9] But at least in some instances, other data protection mechanisms can also qualify for safe harbor. This is most obvious in those statutes that do not use the term encryption, or that also make reference to redaction or other measures that render personal information unreadable or unusable.[10]

---

[4] *See, e.g.*, N.Y. Gen. Bus. Law § 899-AA (2016) (defining "private information" as "personal information consisting of any information in combination with any one or more of the following data elements, when either the personal information or the data element *is not encrypted, or encrypted with an encryption key that has also been acquired*").

[5] *See, e.g.*, Ariz. Rev. Stat. § 18-545(L)(3) (2017).

[6] *See, e.g.*, Alaska Stat. § 45.48.090.

[7] *See* D.C. Code § 28- 3851(1) (2017) ("Acquisition of data that has been rendered secure, so as to be unusable by an unauthorized third party, shall not be deemed to be a breach of the security of the system.").

[8] *See, e.g.*, Wis. Stat. § 134.98(b) (2017).

[9] *See, e.g.*, *supra* notes 4-8.

[10] *See, e.g.*, *infra* note 53.

But even where safe harbor is only provided for encrypted data it is possible that other protection mechanisms can qualify, especially where the term "encryption" is defined broadly.

While encryption generally qualifies for safe harbor, encryption is not the same as hashing, nor are all hashing techniques equally secure. For the eight jurisdictions that consider a username and password, in some form, to constitute personal information, it is not always clear whether hashing qualifies for safe harbor, or, to the extent hashing *can* qualify for safe harbor, whether all or only some hashing processes can do so. Moreover, as discussed below, there are additional considerations with respect to an attacker's ability to crack passwords than the hash function used (i.e. the auxiliary hashing practices discussed in Section IV).

II. OVERVIEW OF CRYPTOGRAPHIC HASHING

Because encryption is strongly associated with breach notification safe harbor provisions, and hashing is often mistakenly equated with encryption, it is important to highlight the key differences between the two. Understanding these differences, as well as some of the limitations of hashing, is critical to the discussion later in this article concerning whether hashing can qualify for safe harbor under state breach notification laws.

The National Institute of Standards and Technology ("NIST") defines encryption as the "conversion of plaintext to ciphertext through the use of a cryptographic algorithm."[11] Importantly, encryption is bi-directional – that

[11] Glossary of Key Information Security Terms, NISTIR 7298 at 69 (2013), http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf.

is, through the operation of cryptographic processes, plaintext (such as the word "hat") can be turned into ciphertext (i.e., encrypted output), and then decrypted back into plaintext using the appropriate cryptographic key.[12] Encryption can be *symmetric*, meaning that the sender and recipient each have a key that both encrypts plaintext and decrypts ciphertext, while *asymmetric* (or "public key") encryption means that each party to a communication has a public key used to encrypt plaintext and a private key used do decrypt ciphertext.[13] Examples of popular encryption algorithms in wide use today are the Advanced Encryption Standard (AES) and the Rivest, Shamir, Adleman (RSA) algorithm.[14]

Hashing, unlike encryption, is intended to be a unidirectional transformation of plaintext (or a "preimage") into a hash value (also known as a "message digest").[15] Hashing algorithms take inputs of plaintext of an arbitrary length and produce alphanumeric digests or "hashes" of a fixed length.[16] For instance, the MD5 hash of "hat" is "46b5e59b2fd342bf8fee10c561958725," while the MD5 hash of "supercalifragilisticexpialidocious" is "08206e04e240edb96b7b6066ee1087af." Hashes are often used to represent sensitive data in a database so that the actual sensitive data does not itself need to be stored because hashes are not meant to be reversed and should, if effective,

---

(citing FIPS 185).

[12] *See id*. at 57 (defining cryptography, citing FIPS 191).

[13] *See id*.

[14] *E.g.*, OWASP, Cryptographic Storage Cheat Sheet at 2.1.2.1, (2017) https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet.

[15] *See* NISTIR 7298, *supra* note 11 at 53 (defining "cryptographic hash function").

[16] *See id.*

create a unique digest for every input.[17] This is particularly useful where it is not actually important *what* the sensitive data is, such that restoring the original plaintext is unnecessary. For instance, hashes can be used to prove that two strings are the same without comparing the strings. Take passwords: when a password is first registered, a hash is generally created and stored in association with the username. When a user attempts to authenticate later, the password entered is run through the same hashing process and the digest is compared to the one on file. If they are the same, the user has, at least in theory, entered the correct password.[18] Hashes are used in many other contexts too, including to prove that a string or file has not changed over time, *i.e.* that it has retained its integrity, as doing so would result in a different digest.[19] They can also be used to identify

[17] *See id.* (noting that an *approved* cryptographic hash function must be one-way and collision resistant. NIST defines collision resistance as "an expected property of a hash function whereby it is computationally infeasible to find a collision," where "collision" is defined as "an event in which two different messages have the same message digest." Quynh Dang, *Recommendation for Applications Using Approved Hash Algorithms*, Special Publication 800-107 at 3 (2012)).

[18] Some hash functions are vulnerable to collisions, such as MD5 and SHA-1, meaning that two different inputs can result in the same hash. While collision vulnerabilities are beyond the scope of this article, it is important to note that hash functions are *intended* to create unique digests for all conceivable inputs. While collision vulnerabilities are cause for significant concern, they present less of a risk in the context of password hashing and more in the context of digital signatures, where the ability to trust that a digital signature is unique is integral to the ability to trust that the entity presenting that signature is, in fact, that entity.

[19] In theory, even a change of one byte in a plaintext input should result in a completely new and different ciphertext after being run through a hashing function. *See, e.g.*, KAREN KENT et al, GUIDE TO INTEGRATING FORENSIC TECHNIQUES INTO INCIDENT RESPONSE, Special Publication

a file or string. For instance, the hash of a photograph can be used to identify instances of that photograph in a data set.[20]

Password hashing algorithms come in a wide variety of types, but we tend to assess their security using a small number of criteria. One of the primary criteria is slowness (also referred to as "expense").[21] Hash algorithms can compute hashes very quickly or slowly depending on their design and configuration. Although there are instances in which a fast hashing algorithm is desirable (such as for computing a hash for an entire disk image, or for creating a checksum to be used to verify the integrity of a file), when it comes to passwords it is desirable for a hashing algorithm to be slow, or expensive.[22] The reason for this is that if an attacker were to obtain the entire password table, we would want to reduce the number of guesses the attacker is able to make each second to the greatest extent possible. MD5 and SHA-1 are two popular hashing algorithms that are generally thought of as being *fast*.[23] On the other hand, bcrypt, and

---

800-86 at 4-8 (2006) (describing the use of hash functions to verify the integrity of copied data during forensic investigations).

[20] For instance, PhotoDNA is a technology that enables companies like Google to identify child pornography images in users' email and other accounts by comparing hashes of images stored in those accounts to "robust hashes" of known pornographic images. *See, e.g.*, Microsoft, "New Technology Fights Child Porn by Tracking Its 'PhotoDNA,'" (Dec. 15, 2009) https://news.microsoft.com/2009/12/15/new-technology-fights-child-porn-by-tracking-its-photodna.

[21] A related criterion is resistance to GPU attacks, discussed below.

[22] *See, e.g.*, NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), Special Publication 800-63B, DIGITAL IDENTITY GUIDELINES: AUTHENTICATION AND LIFECYCLE MANAGEMENT at 15 (2017) ("A memory-hard function SHOULD be used because it increases the cost of an attack.").

[23] *See, e.g.*, Automation Rhapsody, "MD5, SHA-1, SHA-256 and SHA-512 speed performance," (Oct. 23, 2017), https://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-

PBKDF2 are generally thought of as being *slow*.[24] The slower the hashing algorithm, the more computationally expensive it is for the attacker and the less likely that the hashes will be subject to a brute force attack (in which all possible combinations of characters are attempted until the correct password is guessed) or other complex attacks.

III. OVERVIEW OF PASSWORD CRACKING

This section reviews how password cracking works, as this practice illustrates the broad spectrum of (in)secure practices that can be used when creating hashes of passwords. Depending on how a password is hashed, it can be easily cracked in a fraction of a second, rendering the hashing pointless from a security perspective. On the other hand, some hashes could take years to crack, rendering them highly secure in the event of a breach. Accordingly, the type of hashing practices employed by an organization has a direct bearing on whether a particular data breach can qualify for safe harbor.

A full analysis of the current state, and the underlying mechanics, of password hash cracking is beyond the scope of this article.[25] However, some background information is necessary to inform the analysis of the state breach notification statutes discussed below. This article assumes

---

performance.

[24] *See, e.g.*, Bill's Security Site, "Hash Speed Test," https://asecuritysite.com/encryption/htest (last visited Jan. 23, 2018).

[25] For two very useful overviews of these topics, *see* Nate Anderson, "How I became a password cracker," ARS TECHNICA (Mar. 24, 2013), https://arstechnica.com/security/2013/03/how-i-became-a-password-cracker/, & Dan Goodin, "Why passwords have never been weaker—and crackers have never been stronger," ARS TECHNICA (Aug. 2, 2012), https://arstechnica.com/security/2012/08/passwords-under-assault/.

all password cracking would occur (1) offline and (2) in the context of a hacker obtaining an entire database of password hashes. This assumption is generally in line with today's website security practices. It is possible to attempt to crack passwords online, however security controls such as limiting the number of incorrect authentication attempts before an account becomes locked, CAPTCHA requirements, and forcing users to wait an increasingly long period of time between failed log-in attempts have become standards among websites, which prevents many of the attack types discussed below from being usable in this context.[26]

There are several popular methods of cracking password hashes, (*i.e.* of guessing the correct input plaintext and thereby replicating a given hash). Perhaps the most famous of these methods is brute force. While brute force attacks can be infeasible in many contexts, their utility depends in large part on the type and configuration of the hashing algorithm employed by the hash database's owner. For instance, in a 2012 article on the blog *Coding Horror*, the author described having a PC employing two graphics processor units (GPUs) – essentially hyper-powerful processing units designed to handle the intense graphical needs of some computer video games, which are quite popular among password cracking aficionados. Using this PC, the author could create all possible hashes for 6 character passwords hashed with standard MD5 and using uppercase and lowercase letters, numbers, and all possible symbols, in 47 seconds.[27] With only uppercase, lowercase, and numbers in the passwords, the PC in question could

[26] *See* Special Publication 800-63B, *supra* note 22 at 25-26 (discussing recommended controls to protect against online guessing attacks).
[27] Jeff Atwood, "Speed Hashing," *Coding Horror* (Apr. 6, 2012), https://blog.codinghorror.com/speed-hashing/.

create all possible hashes in 3 seconds.[28] As the minimum length and complexity of passwords increase, however, so does the time required to brute force the passwords. For example, the *Coding Horror* author noted at the time of writing that the time required to brute force all 8 character MD5 passwords required to use at least one each of uppercase, lowercase, numbers, and all possible symbols would take approximately 465 days.[29] The time this task would take has likely significantly decreased five years later, in accordance with Moore's law. Moreover, even in 2012 it was possible to create highly specialized, hyper-fast rigs designed specifically for password cracking that essentially daisy chain GPUs. Dan Goodin of *Ars Technica* described one such rig in 2012 that combined 25 GPUs and was capable of trying "180 billion combinations per second against the widely used MD5 algorithm."[30] Today, systems with high powered GPUs can be rented from Amazon Web Services, making these types of attacks more accessible.[31] These high powered, easily accessible systems also facilitate password-based attacks like reverse brute force attacks, also known as password spraying attacks, in which a single, commonly used password is used to attempt to authenticate to a large number of user accounts in fast succession.[32]

---

[28] *See id*.
[29] *See id*.
[30] Dan Goodin, "25-GPU cluster cracks every standard Windows password in <6 hours," ARS TECHNICA (Dec. 9, 2012), https://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/.
[31] *See, e.g.*, Rockfish Sec, "GPU Based Password Cracking with Amazon EC2 and oclHashcat,", Last accessed January 23, 2018 from: http://www.rockfishsec.com/2015/05/gpu-password-cracking-with-amazon-ec2.html.
[32] *See, e.g.*, Jacob Wilkin, "Simplifying Password Spraying," *SpiderLabs Blog*, https://www.trustwave.com/Resources/SpiderLabs-

The attack of choice for password crackers in most cases is the dictionary attack, in which a word list – often times a dictionary or list of passwords recovered from previous data breaches, such as the famed "rockyou.txt" file containing approximately 14.5 million unique preimage[33] passwords recovered from a 2009 data breach – is used to hash guessed combinations of words, numbers, and special characters using predefined rule sets until hashes are generated that match those in a particular hash table.[34] These types of attacks are often carried out using popular, easily accessible freeware such as HashCat and John the Ripper.[35] In years past, a specialized attack using a so-called "rainbow table," a massive, precomputed table for reversing cryptographic hash functions, was viewed as a particularly significant risk in the context of offline password cracking attacks. However, the expense of maintaining such enormous files (often in the range of terabytes in size), the increasing popularity of security features such as salts, and the increasing viability of hyper-fast, increasingly affordable dictionary attacks using daisy-chained GPUs has apparently brought this attack method out of fashion.

Blog/Simplifying-Password-Spraying/ (last visited Jan. 23, 2018).

[33] A preimage is defined as "a message *X* that produces a given message digest when it is processed by a hash function." NISTIR 7298, *supra* note 11 at 5.

[34] *See, e.g.*, ANDERSON, *supra* note 25.

[35] *See id*. These tools will inevitably become even more effective, especially when combined with technologies such as machine learning. *See, e.g.*, Briland Hitaj et al, "PassGAN: A Deep Learning Approach for Password Guessing," *arXiv*:1709.00440 at 2 (Sep. 1, 2017) (describing a method for replacing human-generated password rules with machine learning-generated ones, which when initially tested in combination with HashCat resulted in an 18-24% increase in effectiveness vis-à-vis HashCat alone).

Salts are strings of characters automatically appended to a plaintext password prior to hashing, which add additional complexity and randomness to input plaintext. They are often stored along with each password hash, as the salt used in combination with a plaintext password input is necessary whenever a user attempts authentication. Because the salt is generally randomly generated, sufficiently long (e.g., 128 bytes in bcrypt), and unique to each hash, rainbow tables are generally ineffective when salts are used. A salt that is stored separately from password hashes that is meant to be kept secret is sometimes referred to as a "pepper." NIST currently recommends the use of both a salt and a pepper when hashing passwords.[36]

Depending on the technically enforceable rules an organization has put in place with respect to password complexity, offline brute force and dictionary attacks executed through programs like HashCat generally result in *some* hashes being cracked in a large password hash database, regardless of the hash function used. This appears generally to be a function of the poor quality of the passwords that end up being cracked. For instance, passwords that utilize a word in combination with a number at the end – for example, "picnics7" – will often be trivial to crack because they use a dictionary word and fall under one of several classic password configurations (i.e. a word followed by a single digit, regardless of the hash function used). Even worse, some users employ passwords like "password" and "123456" that can be cracked in milliseconds, as they are among the 20 or so that are most commonly found in password hash tables.

---

[36] *See* National Institute of Standards and Technology (NIST), Special Publication 800-63B, *supra* note 22 at 15, *Digital Identity Guidelines: Authentication and Lifecycle Management* at 15 (2017).

Unless an organization prohibits and technically prevents the use of passwords that are dictionary words, appear on known word lists, conform to common patterns, are prone to brute force attacks (e.g., 000000), or otherwise are vulnerable to known cracking attacks, it is highly likely that if a password hash database is acquired by bad actors, some of the hashes will be cracked. NIST guidance provides that, prior to accepting a password, a verifier should compare it "against a list that contains values known to be commonly-used, expected, or compromised" including "passwords obtained from previous breach corpuses, dictionary words, repetitive or sequential characters (e.g. 'aaaaaa', '1234abcd'), [and] context-specific words, such as the name of the service, the username, and derivatives thereof."[37]

PBKDF2 and bcrypt are well regarded hash functions for passwords, in particular, because they are far less vulnerable (meaning they offer greater "preimage resistance") to attacks utilizing GPU daisy-chaining.[38] For instance, following Ashley Madison's highly publicized data breach in 2015, in which 36 million hashes were exfiltrated from Madison's servers, the owner of the famed 25-GPU cracking rig declined to even attempt to crack the hashes in the leaked table explicitly because Madison had used bcrypt.[39] Indeed, the owner of the specialized rig "estimated

---

[37] *Id*. at 14.

[38] Preimage resistance is defined as "an expected property of a hash function such that, given a randomly chosen message digest, *message_digest*, it is computationally infeasible to find a preimage of the *message_digest*. NISTIR 7298, *supra* note 11 at 5. Preimage resistance is "measured by the amount of work that would be needed to have a high probability of finding a preimage for a hash function." *Id*. at 7.

[39] Dan Goodin, "Lessons learned from cracking 4,000 Ashley Madison passwords," ARS TECHNICA (Aug. 26, 2015), https://arstechnica.com/security/2015/08/cracking-all-hacked-ashley-

it would take years using a highly specialized computer cluster just to check the dump for the top 10,000 most commonly used passwords."[40] Nonetheless, another researcher stepped in to try his hand at cracking some of the Ashley Madison password hashes, and "after five days of nonstop automated guessing using a moderately fast server specifically designed to carry out compute-intensive cryptographic operations, he deciphered just 4,000 of the underlying plaintext passwords."[41] Among the cracked passwords were "123456," "password, "12345," qwerty", and some explicit phrases. While 4000 cracked hashes over four days is generally disappointing for researchers in the context of a leak of more than 30 million leaked hashes, it is important to emphasize that even where one of the most highly recommended hashing functions is used, at least some leaked or stolen hashes are likely to be cracked in the immediate aftermath of a data breach unless the impacted organization has a particularly strong set of controls in place governing password construction.

There is at least one other potentially significant risk associated with password hashing that is sometimes overlooked in discussions of its use. While the use of an expensive hashing algorithm in combination with a strong password policy can be immensely helpful in combatting password cracking when a bad actor obtains an entire password table, their effectiveness may be reduced in some instances if an attacker is targeting a specific user. For instance, if an attacker combines social engineering or open source research on an individual known to be included in a password hash dump, he or she may be better able to crack

madison-passwords-could-take-a-lifetime/.

[40] *Id.*

[41] *Id.*

the single hash associated with that individual.[42] This risk is especially heightened where an individual uses the same password at multiple websites.[43] The common availability of password lists sourced from large scale data breaches facilitates attackers' ability to target individual users.[44]

IV. SUMMARY OF KEY CONSIDERATIONS FOR SECURE PASSWORD HASHING

As the previous sections discussed, both hashing and encryption are cryptographic in nature, but they are not the same thing, and they have different use cases. The differences between encryption and hashing are important to understand from a policymaking perspective. Indeed, viewing the two as interchangeable could lead to breach notification statutes that offer protections to entities that

---

[42] For instance, in an August 2017 blog post, "the hacker known as 'Alex'" describes attempting to gain access to a friend's email account as part of an authorized white hat exercise using a multitude of publicly available resources and data, including social media and publicly available lists of passwords and password hashes associated with known data breaches, many of which can be researched using the well-intentioned https://haveibeenpwned.com. *See* The hacker known as "Alex," "Operation Luigi: How I hacked my friend without her noticing," https://defaultnamehere.tumblr.com/post/163734466355/operation-luigi-how-i-hacked-my-friend-without (last visited Feb. 18, 2018).

[43] *See* Martin Bos, "Introduction to GPU Password Cracking: Owning the LinkedIn Password Dump," *Trusted Sec Update* (Jun. 17, 2016), https://www.trustedsec.com/june-2016/introduction-gpu-password-cracking-owning-linkedin-password-dump ("The LinkedIn list offers an opportunity for us . . . .[to] show how we attack large password breach lists. The passwords gained from these types of breaches are very valuable to us on penetration tests because people often reuse passwords across work and social media.").

[44] *Id.*

suffer data breaches in spite of their use of weak, easily cracked password hashes. This outcome would be contrary to the goals of breach notification statutes.

As discussed in the previous sections, it is apparent that the hash function that is used to hash passwords is a key consideration, and that the expensiveness of the function is a primary criterion for selecting a hash function. However, in addition to the function selected, there are a number of other practices that can be combined with the hash function to maximize the chances of protecting all hashed passwords in the event of a data breach. These include the use of an appropriate salt, the use of a pepper, and the use of password selection rules that prevent the use of passwords that are easy to guess, such as those found in known breach corpuses and the dictionary, or that would otherwise be easily cracked using programs like HashCat. This article will sometimes refer to these practices as "auxiliary hashing practices." Even if a strong hash function is used to hash passwords, a data breach involving password hashes that were not subject to most (if not all) of these auxiliary hashing practices is likely to lead to at least some of the password hashes being cracked.

V. PASSWORD HASHES IN THE CONTEXT OF STATE BREACH NOTIFICATION REQUIREMENTS

As noted previously, there are several variations on breach notification safe harbor provisions, including those that refer to encryption, as well as those that refer to additional or alternative mechanisms for protecting data.[45] This Article focuses on the safe harbor provisions in the statutes of eight states that consider some form of a user ID

[45] *See supra* Section I.

and password to be "personal information."

In most contexts, safe harbor provisions in state breach notification statutes are sufficient for their intended purpose and will usually be satisfied using generally accepted types of encryption (e.g., AES 256). Most breach notification statutes are primarily concerned with breaches affecting data such as Social Security numbers and financial account numbers, which can be protected with encryption that easily meets the applicable statutory requirements. However, the same is not necessarily true for passwords, which are generally protected through hashing.

As described in greater detail below, the hashing of passwords at least arguably qualifies for safe harbor for purposes of the eight state breach notification statutes discussed in this article, although in some of the statutes the argument is more difficult to make. In fact, in some states the legislature may not have intended for password hashing to be subject to the special protections afforded to traditional encryption under the state breach notification laws. For instance, the Rhode Island statute refers to the "transformation of data through the use of a one hundred twenty-eight (128) bit or higher algorithmic process into a form in which there is a low probability of assigning meaning without use of a confidential process or key," which would only make sense in the context of a small subset of hash functions. When Rhode Island was in the process of amending its breach notification statute in 2015 to include this definition of encryption, several commenters pointed out to the legislature that the definition was overly narrow, but the legislature did not change the definition of encryption in response to these calls, which suggests that it did not agree with the commenters.[46]

[46] For instance, one trade association pointed out that "'128 bits' may

In some states, hashing may not *originally* have been intended to qualify for safe harbor, as the inclusion of usernames and passwords in the statutory definition of personal information occurred after the inclusion of a safe harbor provision.[47] For instance, Illinois had already included an encryption safe harbor provision when it passed HB1260, which added login credentials to the definition of "personal information," in 2016. [48] Nevada's statute also already provided safe harbor for encrypted data when it added login credentials to the definition of personal

not be possible or appropriate depending on the data and systems being utilized Letter from Laura Dooley, Auto Alliance, to the Honorable Brian Patrick Kennedy (Mar. 17, 2015), (on file with the author). Another commenter pointed out that the "exact method of encryption is less important than that the information is rendered unreadable or unusable, which is the standard used by many other states." Letter from James J. Halpert, State Privacy & Security Sec. Coal., Inc., to the Honorable Brian Patrick Kennedy (Mar. 17, 2015), (on file with the author).

[47] This is due to the nature of hashing and the traditional types of data sought to be protected by data breach notification laws. For example, Social Security numbers consist of nine digits and could therefore be relatively easy to brute force if hashed without using additional security measures. Driver's license numbers and credit card numbers would likely also be relatively easy to brute force if the pattern used were to be known, or if known prefixes, such as Bank Identification Numbers, were to be used. *See, e.g.*, Simson L. Garfinkel, "De-Identification of Personal Information," NISTIR 8053, at 17 (2015), https://nvlpubs.nist.gov/nistpubs/ir/2015/NIST.IR.8053.pdf (describing brute force attack against taxi medallion numbers that were hashed and publicly released in a dataset).

[48] *See* HB1260 (Ill. 2016), http://www.ilga.gov/legislation/publicacts/99/PDF/099-0503.pdf.

information in AB179 in 2015.[49] Nebraska[50] and Wyoming[51] also added user credentials to the definition of personal information after their current safe harbor provisions already existed in their respective statutes. Conversely, California added its current definition of encryption after it added user credentials to the definition of personal information.[52]

Regardless of the original legislative intent, several of the state breach notification statutes discussed below appear to implicitly acknowledge that hashing can qualify for safe harbor by providing a sufficiently broad exemption for notification that is not directly tied to encryption.[53] But even where hashing can in theory qualify for safe harbor, the question of whether it constitutes "encryption" or otherwise

---

[49] *See* A.B. 179 (Nev. 2015), http://www.leg.state.nv.us/Session/78th2015/Bills/AB/AB179_EN.pdf.
[50] *See* L.B. 835 (Neb. 2016), https://nebraskalegislature.gov/FloorDocs/104/PDF/Slip/LB835.pdf.
[51] *See* S. File 0036 (Wis. 2015), http://legisweb.state.wy.us/2015/Engross/SF0036.pdf.
[52] *See* S.B. 46 (Cal. 2013), https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201320140SB46 (adding user credentials to definition of personal information); A.B. 964 (Cal. 2015), https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201520160AB964 (defining encryption).
[53] See, e.g., Cal. Civ. Code § 1798.82 (2017) (defining the term "encryption"); Fla. Stat. § 501.171 (2017) (providing notification safe harbor for data that is "secured or modified by any other method or technology that removes elements that personally identify an individual or that otherwise renders the information unusable"); Neb. Rev. Stat. § 87-802 (2017) (defining the term "personal information" to exclude records where either the name or data elements are, among other things, "otherwise altered by any method or technology in such a manner that the name or data elements are unreadable"); 10 L. of P.R. § 4051 (2017) (providing safe harbor where data is not legible without the use of a special cryptographic code); Wyo. Stat. § 40-12-501(2016) (defining the term "redaction").

qualifies for notification safe harbor in any specific instance will usually require a case-specific, context-driven analysis. Moreover, it is always possible that regulators, plaintiffs, and other groups inclined to construe breach notification laws could conservatively argue that only hash functions that are sufficiently expensive, in combination with implementation of the auxiliary hashing practices discussed previously, can qualify for safe harbor.

None of the statutes discussed in this section explicitly address hashing or any of the auxiliary hashing practices, even though NIST offers detailed guidance on these issues.[54] Instead, people on both sides of the debate are left wondering what hashing practices are sufficient for hashing, for example, to render personal information inaccessible, or "unusable, unreadable, or indecipherable."

As of the time of writing, the jurisdictions that consider online account credentials to be personal information include California, Florida, Illinois, Nebraska, Nevada, Puerto Rico, Rhode Island, and Wyoming.[55] The safe harbor provisions in these statutes fall into three primary categories: (1) safe harbor applies where data is encrypted;[56] (2) safe harbor applies where the data is encrypted or protected in some alternate manner; and (3) safe harbor

[54] *See* Special Publication 800-63B, *supra* note 22, at 13–15.

[55] Maryland and Delaware recently passed laws updating their breach notification statutes, which, once effective, will add similar data elements to their respective definitions of personal information. *See* H.B. 974 (Md. 2017), http://mgaleg.maryland.gov/2017RS/bills/hb/hb0974E.pdf; House Substitute 1 for H.B. 180 (Del. 2017), https://legis.delaware.gov/BillDetail/26009. This article does not analyze the requirements in those bills.

[56] Some statutes also condition safe harbor on an encryption key not being compromised, but for the purposes of this article it is assumed that there has been no compromise of such keys.

applies where the data is protected using a mechanism that does not specifically refer to encryption. Within each of these categories, there are statutes that (a) provide definitions for encryption or alternate data protection mechanisms, and (b) those that do not. After examining the safe harbor provisions in these states' statutes, this article finds that hashing could, in theory, qualify for safe harbor in all eight states, but that in some of the statutes there are ambiguities or other issues that make this argument more difficult to make – most notably in Rhode Island and Nevada, and to a lesser extent in Illinois. Moreover, even where hashing can more easily qualify for safe harbor in theory, whether it qualifies in practice likely depends on the hashing algorithm used and whether any auxiliary hashing functions were used.

A. Safe Harbor Applies Where Data is Encrypted, and the Term is Defined

California: California's statute would generally consider hashing to be encryption if it rendered personal information "unusable, unreadable, or indecipherable to an unauthorized person through a security technology or methodology generally accepted in the field of information security."[57] There is little question that hashing is able to render plaintext "unusable, unreadable, or indecipherable to an unauthorized person." The next question is whether the hash functions already discussed here are "generally accepted in the field of information security." Some commentators, including a former Chief Technologist of the FTC, suggest that the inexpensive nature of SHA-1 and MD5 is not an intrinsic vulnerability in those functions. Rather,

[57] Cal. Civ. Code § 1798.82(i)(4).

these commentators argue that a greater number of iterations should be used to increase the preimage resistance of fast hash functions.[58] Indeed, the ability to increase the "cost," *i.e.*, the number of iterations, is a built-in feature of bcrypt that makes it particularly useful in the face of Moore's law.[59] Increasing the iterations of MD5 or SHA-1 would follow the same logic.[60] Other commentators argue that only hash functions that specifically provide resistance against GPU attacks should be used, such as bcrypt (which allegedly resists GPU attacks better due in part to its incorporation of the Blowfish block cipher). According to one well known security researcher:

> Blowfish is a block cipher with a notoriously expensive setup time. To optimize Blowfish to run much faster, you'd have to contribute a major advance to cryptography. We security practitioners are all betting people, and we usually like to place our bets on the side that demands major advances in cryptography. [The

[58] *See* Steven Bellovin, *Storing passwords, or the risk of a no-salt diet*, TECH@FTC BLOG (Mar. 21, 2013), https://www.ftc.gov/news-events/blogs/techftc/2013/03/storing-passwords-or-risk-no-salt-diet ("For password storage, any hash function for which one cannot compute preimages is appropriate; you just have to set the iteration count properly.").

[59] NIELS PROVOS & DAVID MAZIÈRES, *A Future-Adaptable Password Scheme*, PROCEEDINGS OF THE FREENIX TRACK: 1999 USENIX ANNUAL TECHNICAL CONFERENCE (1999), https://www.usenix.org/legacy/event/usenix99/provos/provos.pdf (last visited Jan. 23, 2018).

[60] Conversely, at least one court has refused to dismiss a claim that SHA-1 is "below the 'bare minimum' security practice" with regard to the hashing of passwords." *In Re LinkedIn User Privacy Litig*., Case No.: 5:12-CV-03088-EJD at 15 (N.D. Cal. Mar. 28, 2014).

> creators or bcrypt also] extended Blowfish. They call theirs 'Eksblowfish.' Eksblowfish is pessimized: the setup time takes even longer than Blowfish. How long? Your call. You can make a single password trial take milliseconds, or you can make it take hours.[61]

Because of this innate resistance to GPU attacks, the argument goes that using bcrypt makes far more sense than a highly iterated MD5 or SHA-1.

NIST has also weighed in on this debate in the context of advising the federal government on its own security practices. For instance, NIST recommends the use of PBKDF2 as a key derivation function,[62] which is similar to hash function but used to derive cryptographic keys based on input passwords. Further, in its Special Publication 800-63B, *Digital Identity Guidelines Authentication and Lifecycle Management*, NIST states that "memorized secrets SHALL be salted and hashed using a suitable one-way key derivation function" and specifically lists, as examples, PBKDF2 and Balloon.[63] Moreover, as discussed further

---

[61] Thomas Ptacek, *Enough With The Rainbow Tables: What You Need To Know About Secure Password Schemes*, MATASANO SECURITY (Sep. 7, 2007), https://web.archive.org/web/20130407190430/http://chargen.matasano.com/chargen/2007/9/7/enough-with-the-rainbow-tables-what-you-need-to-know-about-s.html.

[62] NIST, Special Publication 800-132, RECOMMENDATION FOR PASSWORD-BASED KEY DERIVATION, PART 1: STORAGE APPLICATIONS at 7 (2010).

[63] Special Publication 800-63B, *supra* note 22 at 15. NIST further states that "[t]he key derivation function SHALL use an approved one-way function such as Keyed Hash Message Authentication Code (HMAC) [FIPS 198-1], any approved hash function in SP 800-107, Secure Hash Algorithm 3 (SHA-3) [FIPS 202], CMAC [SP 800-38B] or Keccak Message Authentication Code (KMAC), Customizable

below, NIST has weighed in on this topic in the context of its Federal Information Processing Standards. However, NIST's opinion does not take into consideration the existence of bcrypt.

Accordingly, while hash functions like bcrypt and PBKDF2 appear to be consensus picks as "generally accepted in the field of information technology," an argument can likely also reasonably be made that fast hash functions like MD5 and SHA-1 meet this criterion provided that an appropriate number of iterations is used to increase the preimage resistance of these hashes to GPU attacks.

Rhode Island: Rhode Island defines encryption as "the transformation of data through the use of a one hundred twenty-eight (128) bit or higher algorithmic process into a form in which there is a low probability of assigning meaning without use of a confidential process or key."[64] The reference in the statute of a requirement that the algorithm used in encryption be 128-bit or higher creates ambiguity in the context of hashing. The reference makes sense in the context of traditional encryption, where longer key lengths generally, though not always, correspond to greater difficulty in cracking ciphertext. In the context of hashing, however, it is not clear what a "128-bit or higher algorithmic process" might be. Although some types of hashing, such as the hashed message authentication code ("HMAC"), do use

SHAKE (cSHAKE), or ParallelHash [SP 800-185]." *Id.*

[64] R.I. Gen. L. § 11-49.3-3(2) (2017).

a key,[65] many commonly used hash functions do not.[66]

This is not to say that the statute's definition of encryption entirely precludes the application of safe harbor to hashing. One possibility is that 128-bits refers to the length of the hash itself, which is generally 128 bits or longer in the hash functions discussed in this article. It could also potentially refer to keys that are sometimes used internally by hash functions in the hashing process. However, if it is the latter, it may be difficult for some hash functions to satisfy this requirement. For instance, bcrypt, which relies on Blowfish, a symmetric key block cipher, uses keys that range between 32 and 448 bits.[67] But MD5 does not use keys at all.[68]

In light of the problematic definition of encryption in Rhode Island's data breach notification statute, it is more difficult to argue that hashed passwords qualify for safe harbor in that state. The Rhode Island legislature should strongly consider clarifying this issue, given its decision to include usernames and passwords in the definition of personal information and the reality that most entities hash

---

[65] *See* FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 198-1, THE KEYED-HASH MESSAGE AUTHENTICATION CODE (HMAC), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf (last visited Jan. 23, 2018). According to RFC 2014, "HMAC: Keyed-Hashing for Message Authentication," keys used in HMAC should be at least as long as the byte-length of hash outputs. The byte-length for MD5 is 16, which is 128 bits. The byte output for SHA-1 is 20, or 160 bits. *See* H. KRAWCZYK et. al., HMAC: KEYED-HASHING FOR MESSAGE AUTHENTICATION (1997) https://tools.ietf.org/html/rfc2104 (last visited Jan. 23, 2018).

[66] This is true for all of the algorithms discussed in this article.

[67] *See* Bruce Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, in FAST SOFTWARE ENCRYPTION, CAMBRIDGE SECURITY WORKSHOP PROCEEDINGS 191(1993).

[68] *See generally* RFC 1321, https://tools.ietf.org/html/rfc1321.

passwords. However, given the legislative history discussed above, it may be that the Rhode Island legislature is not interested in broadening the definition of encryption despite the issues identified in this article.

B. Safe Harbor Applies Where Data is Encrypted, and the Term is Not Defined

Nevada: Nevada's statute provides safe harbor where the name and data elements are encrypted, but it does not define encryption. Notably, another, closely related Nevada statute that addresses data security does. Specifically, NRS 603A.215 defines encryption as:

> The protection of data in electronic or optical form, in storage or in transit, using… an encryption technology that has been adopted by an established standards setting body . . . which renders such data indecipherable in the absence of associated cryptographic keys necessary to enable decryption of such data.

The definition of encryption in Nevada's data security statute could be referenced by a court, plaintiff, or regulator to interpret the use of that term in the state's breach notification statute. There are, however, a number of challenges to doing so, including that (1) the Section 215 definition is not referenced in the state's breach notification statute and (2) the Section 215 definition is specifically limited to "as used in this section," i.e. Section 215. The primary argument in favor of looking to the data security statute's definition is that the two statutes are in the same chapter of the Nevada code, in close proximity to each other, and that the term is undefined in the breach notification

statute.

Assuming that Section 215 would at least be considered in interpreting Nevada's beach notification statute, it is worthwhile to walk through that analysis. While Section 215's definition of encryption appears to be intended to describe encryption, not hashing, the definition could potentially apply to a hash function, primarily because hash functions generally render preimages indecipherable, albeit without a key to reverse the process. Hash functions could loosely fall under the umbrella of "encryption technology" because they are cryptographic in nature, even if hashing and encryption are technically distinguishable. At the same time, as this article makes clear, hashing is distinct from encryption, so the use of the term "encryption technology" could easily be interpreted to preclude hashing. As a result, application of the Section 215 definition could make it difficult for hashing to qualify for safe harbor.

Another question that must be answered under the Section 215 definition in the context of hashes is that the selected hash must be adopted by an established standards setting body, "including, but not limited to, the Federal Information Processing Standards issued by the National Institute of Standards and Technology."[69] The Federal Information Processing Standards, otherwise known as the FIPS, are a series of "standards and guidelines that are developed by the National Institute of Standards and Technology (NIST) for Federal computer systems" pursuant to the Information Technology Management Reform Act (Public Law 104-106).[70] Notable FIPS, for the purposes of

[69] Nev. Rev. Stat. Ch. 603A.215(5)(b).

[70] NIST, FIPS GENERAL INFORMATION, https://www.nist.gov/information-technology-laboratory/fips-general-information (last visited Jan. 23, 2018).

this article, are 140-2, *Security Requirements for Cryptographic Modules*, FIPS 180-4, *Secure Hash Standard (SHS)*, and FIPS 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. NIST has also issued special publications in which they endorse specific hashing algorithms, including Special Publication 800-132, *Recommendation for Password-Based Key Derivation, Part 1: Storage Application*, and Special Publication 800-63B, *Digital Identity Guidelines Authentication and Lifecycle Management*. Other standards setting bodies have likewise endorsed specific hash functions.

If the Section 215 definition is not applied to the undefined term "encryption," it is unclear whether hashed passwords would qualify for safe harbor in Nevada. Because the term is undefined and preexisted the addition of user credentials to the definition of personal information, it is possible to argue that the Nevada legislature overlooked this issue while intending for industry-standard security protections for passwords to be subject to safe harbor. At the same time, one could also easily argue that the term "encryption," by definition, precludes hashing from qualifying for safe harbor. This is an issue that the Nevada legislature should consider addressing as soon as possible.

C. Safe Harbor Applies Where the Data is Encrypted or Protected in Some Alternate Manner, and the Terms are Defined

Nebraska: Nebraska's statute provides safe harbor if either the name or the data elements are "encrypted, redacted, or otherwise altered by any method or technology in such a manner that the name or data elements are

unreadable."[71] Although the statute defines both "encrypted" and "redacted," it is not necessary to analyze whether hashing would satisfy those definitions, as it also carves out from the definition of personal information records where "either the name or the data elements are . . . otherwise altered by any method or technology in such a manner that the name or data elements are unreadable."[72] Hashing would reasonably satisfy this requirement. The only question in this context would be whether a hash that would be trivial to crack would reasonably be considered "unreadable."

D. Safe Harbor Applies Where the Data is Encrypted or Protected in Some Alternate Manner, and the Terms are Not Defined

Florida: Florida's statute provides notification safe harbor for data that is "encrypted, secured or modified by any other method or technology that removes elements that personally identify an individual or that otherwise renders the information unusable."[73] Hashing generally qualifies for this exception. Hash functions that create hashes from passwords arguably render the information unusable. However, whether the data is truly unusable or rather can realistically be cracked likely hinges on the hash function used and whether the auxiliary hashing practices have been implemented, as previously discussed.

Illinois: The Illinois statute provides safe harbor for data that is encrypted or redacted, but it does not define

---

[71] Neb. Rev. Stat. § 87-802(5)(a).
[72] *Id.*
[73] Fla. Stat. § 501.171(g)(2).

either term.[74] Despite the lack of definitions, it is possible that hashing a password could meet the intent of the law, particularly as the safe harbor provision is included specifically in the portion of the definition of personal information that includes user credentials.[75] The same challenges discussed in the context of the Nevada statute would apply to the undefined use of the term "encryption" in the Illinois statute. However, it is easier to argue that hashing qualifies for safe harbor in Illinois in light of the statute's undefined use of the term "redacted." Nonetheless, this is an issue that the Illinois legislature should consider addressing as soon as possible.

E. Safe Harbor Applies Where the Data is Protected Using a Mechanism that Does Not Specifically Refer to Encryption, and the Terms are Defined

Wyoming: Wyoming's statute provides safe harbor where the data elements are "redacted," which is defined as "alteration or truncation of data such that no more than five (5) digits of the data elements . . . are accessible as part of the personal information."[76] Hash functions could potentially fall under this definition. Hash functions alter data such that no digits – let alone five – of the data elements are readable or otherwise usable. At the same time, the use of the term "digits" is ambiguous in this context because a

---

[74] *See* 815 Ill. Comp. Stat. § 530/5 (2017).

[75] This language was added at the same time user credentials were added to the definition of personal information, which may indicate legislative intent for hashing to qualify for safe harbor. *See* HB1260 (Ill. 2016), http://www.ilga.gov/legislation/publicacts/99/PDF/099-0503.pdf.

[76] Wyo. Stat. § 40-12-501(a)(8).

password may consist only of letters and special characters and contain no digits at all. Assuming the term does apply to alphanumeric values, the primary question under this statute would be whether the underlying passwords are "accessible" even though hashed, which in turn likely hinges on the hash function used and whether the auxiliary hashing practices have been implemented, as previously discussed.

It is also worth noting that due to a quirk in the statute, it is not clear if the Wyoming statute provides any safe harbor at all for "redacted" data. This is because the definition of "redact" appears to apply only to "the data elements provided in subparagraphs (vii)(A) through (D) of this subsection," all of which were repealed in 2015. Thus, the definition technically does not apply to the statute as drafted today. This is an issue the Wyoming legislature should address as quickly as possible.

F. Safe Harbor Applies Where the Data is Protected Using a Mechanism that Does Not Specifically Refer to Encryption, and the Terms are Not Defined

Puerto Rico: Puerto Rico's statute requires notification if personal information is "legible enough so that in order to process it there is no need to use a *special cryptographic code*."[77] The term "special cryptographic code" is not defined. Despite the lack of a definition, it seems likely that hashing a password would meet the intent of the law. Hash functions are cryptographic in nature and create an output that is theoretically not usable or readable even when an attacker has possession of the hash, just as with ciphertext. Of course, as discussed previously, it is possible

[77] 10 L. P.R. Ann. § 4051(a).

that a fast hash in combination with a weak password might not be considered to qualify by some judges, regulators, or plaintiffs. Thus, the use of an expensive hash function in combination with the implementation of the auxiliary hashing practices remains an important consideration.

VI. CONCLUSION

As is evident from the above overview, not all hashes are created equal. The "expensiveness" of the function, the presence of a salt (and sometimes a pepper), the function's resistance to GPU attacks, and the organization's password policies, among other things, constitute criteria that must be assessed when determining if a particular use of a password hash function is sufficiently secure to qualify for safe harbor.

Legislators in states that consider usernames and passwords to constitute personal information, or that are considering amending their statutes to do so, should recognize that the vast majority of organizations protect that information through hashing. Accordingly, they should make clear in their states' breach notification statutes that hashing can qualify for breach notification safe harbor as a general matter. Doing so does not require making explicit reference to hashing, provided that the language used, either in a separate definition or in the definition of "personal information," clearly applies to hashing. A definition that would meet this criterion, for example, would provide safe harbor where data elements are "encrypted, redacted, or altered in a manner that renders the elements unusable, unreadable, or indecipherable to an unauthorized person."

Legislators also need to take into consideration that some hashes would not be secure enough to warrant safe harbor. They must therefore consider how best to identify in their states' statutes the minimum acceptable qualifying

criteria for the expensiveness of the hash function, the type of salt and/or pepper, and the auxiliary hashing practices that must also be implemented alongside the selected hash function, out of recognition that notification should not be necessary when passwords are hashed in a sufficiently secure manner. Aside from listing these criteria individually, a statute could potentially include, among other options, (1) a requirement that if a hashing function is used on passwords, the function and password policy used be configured consistent with the Memorized Secret Verifiers section of NIST's Special Publication 800-63B, *Digital Identity Guidelines Authentication and Lifecycle Management*; and/or (2) an option to have an expert certify that a specialized password cracking rig would be unable to crack more than a specific (minute) percentage of the password hashes stored in a table in the event of a breach.