HIPAA Certification of the Provisio De-identification Engine

November 30, 2006

by Fritz Scheuren and Patrick Baier

Contents

1. DECLARATION OF FRITZ J. SCHEUREN AND PATRICK D. BAIER

1. Fritz J. Scheuren is a resident of Alexandria, Virginia, is over the age of 18 and has personal knowledge of the matters set forth in this declaration. Patrick D. Baier is a resident of Washington, D.C., is older than 18 years and has assisted Fritz Scheuren in a number of HIPAA related certifications.

1.1. Qualifications

1.1.1. Fritz Scheuren

2. Fritz Scheuren is the Vice President for Statistics at NORC, a research arm of the University of Chicago, and has been so from 2001 through the present. At the Urban Institute, from 1999 to 2001, he directed the National Survey of America's Families ("NSAF") and made all decisions on how to release NSAF data to the public in a de-identified form. From 1997 to 1999, he served as National Technical Director, Statistical Sampling Economics Group, for Ernst & Young, LLP. From 1994 to 1996, he was a Professor of Statistics at the George Washington University, where he has served as Adjunct Professor of Statistics since 1988. From 1980 to 1994, Dr. Scheuren was Director, Statistics of Income Division, at the Internal Revenue Service. He was Chief Mathematical Statistician of the Social Security Administration from 1978 to 1980.

3. Dr. Scheuren is an expert on the use of generally accepted statistical and scientific principles and methods for de-identifying information - and has even developed some of these methods. He has over thirty years of experience as a mathematical statistician. He received his Ph.D. in Statistics from the George Washington University in 1972. He has published over two hundred applied and theoretical papers, monographs, and books focused on privacy and confidentiality issues, administrative record research, record linkage, survey sample design, and estimation. He is a member of the following professional organizations, among others: American Statistical Association (Vice President, 1999 to

2001, President Elect 2004, President, 2005); International Association of Survey Statisticians (Scientific Secretary, 1997); National Academy of Sciences, Applied and Theoretical Statistics (1994-97); and Washington Statistical Society (President, 1991-92). He served as Associate Editor of the Journal of the American Statistical Association from 1989 to 1996 and remains an Associate Editor of the Survey Methodology Journal, where he has served since 1986.

4. In 1994, Dr. Scheuren was a member of the Federal Committee on Statistical Methodology ("FCSM") and, as such, participated in the drafting of Statistical Policy Working Paper 22, Report on Statistical Disclosure Limitation Methodology. This FCSM volume was published by the Statistical Policy Office, Office of Information and Regulatory Affairs, Office of Management and Budget in May, 1994, as Statistical Working Paper No. 22 ("Working Paper 22")[1] Under the privacy regulations issued pursuant to the Health Insurance Portability and Accountability Act of 1996 ("HIPAA"), the Secretary of Health and Human Services has specifically approved the use of FCSM Working Paper 22 as one of two guidelines to "generally accepted statistical and scientific principles and methods for rendering information not individually identifiable." The second, more process-oriented guide mentioned by the HIPAA Privacy Rule is the FCSM "Checklist on Disclosure Potential of Proposed Data Releases."[2] While Dr. Scheuren did not work on development of this second FCSM protection tool, he has used it extensively and commented on its strengths and limitations in several publications. Dr. Scheuren regularly speaks before statistical meetings regarding privacy matters and methods to protect the confidentiality of individual data. In November 2002, for example, he spoke before the FCSM at their annual government-wide meeting, commenting on new methods to afford adequate de-identification protections in federal government data products.

[1] www.fcsm.gov/working-papers/SPWP_22_rev_TOC.pdf
[2] http://www.fcsm.gov/committees/cdac/checklist.doc

1.1.2. Patrick Baier

5. Dr. Baier holds a D.Phil. in mathematics and brings in his experience both with statistical programming and the use of cryptography in the HIPAA world. He has been working as a statistician for the National Opinion Research Center (NORC) on a number of projects.

6. In several HIPAA certifications over the past three years for a number of health care companies, many of them operating on a national level, Patrick Baier has assisted Fritz Scheuren in carrying out statistical analyses. Moreover, Dr. Baier has reviewed encryption technologies used to de-identify data, including a review of protocols and source code. In 2006, Dr. Baier has spoken about the HIPAA Rule in an invited session at the Joint Statistical Meeting in Seattle.

1.2. Overview

7. This document discusses Provisio's **iTrials Service Platform**, a tool for de-identifying medical data protected under the Health Insurance Portability and Accountability Act (HIPAA) of 1996. The HIPAA Rule stipulates that Covered Entities (essentially health care providers, health insurance companies and health care service clearing houses) may not disclose personally identifiable health information (PHI) to outside parties unless certain privacy protections are in place. These could be either a business associate agreement which would impose certain legal obligations on the data recipient as to the use and handling of patient data, or a "safe harbor" approach whereby certain (potentially) identifying data fields must be removed prior to dissemination of the data, or, finally, a statistician's certification asserting that there is only a very small risk of re-identification from a data set.

8. In most applications, business associate agreements are not a viable option for companies wishing to buy, sell or analyze medical data. Hence, under the other options, data need to be sufficiently de-identified so as to meet the safe harbor or statistician's

standard before they can be given out, e.g., for market analyses, by a Covered Entity.

9. Patient identifiers used to be a valuable pieces of information that would allow the data analyst to link records, often obtained from different sources or over extended periods of time, by patient. Typically, names, birth dates, gender, or addresses would be used to achieve this longitudinal linkage. Because the use of these identifiers is generally no longer permitted under HIPAA, the health care industry has moved to develop technologies to create unidentifiable linking codes which are still (essentially) unique to a patient, similar to the combination of names, birth dates, gender, etc., but which do not allow the user of the data to derive or extract any of these identifying pieces of information. Cryptographic techniques are central to achieving the required security of linking codes, that is, strong cryptography is necessary to construct the linking codes in such a manner that they cannot be reverse engineered.

10. This document discusses an implementation of such a technique by Provisio. We first discuss how the concept of hashed linking codes fits into the HIPAA legislation in Section 2. This discussion reflects our interpretation of the HIPAA Rule, speaking as statisticians and data analysis practitioners. However, we cannot speak as legal experts on this issue. It seems that the legal basis for such hashed linking codes has not yet been tested in the courts, and we are unaware of any discussion of this issue by legal experts. In Section 3, we summarize the important technical details of the iTrials Service Platform. We discuss the soundness and suitability of this technology in Section 4. As in all cryptographic applications, "textbook" cryptography does not always equal "real world" security. It is important that any cryptographic application be embedded into a framework of physical, electronic, procedural safeguards, etc. Industry standard solutions, implemented with due diligence, are typically sufficient in a HIPAA context. We discuss this briefly, without becoming too technical, in Section 5. Our conclusions follow in Section 6.

11. The purpose of this document is solely to investigate the security of Provisio's

iTrials de-identification engine, that is, we assert that it is extremely difficult for an intruder to recover any identifying information from the linking codes *alone* which are created by this engine. It is not within the scope of this document to certify that any particular data set processed through this de-identification engine meets the standard of de-identification required by HIPAA. Whether a data set meets the HIPAA requirements for de-identification depends on what other data elements are kept in the data set, in addition to the linking code. In order to certify a data set to be de-identified as required by HIPAA, a statistician's analysis of the entire data set and the (potentially identifying) information contained in it are needed. This certification document at hand provides only one element in such an analysis, by asserting that the linking codes alone do not pose a more than de minimis risk of re-identification.

2. THE USE OF LINKING CODES UNDER HIPAA

12. There has been some controversy about the use of linking codes under HIPAA, that is, the creation of hashed or encrypted or otherwise irreversible codes or tokens from identifiable patient information. These codes are unique to the patient and can be used for longitudinal record linkage, but at the same time they are incapable of being reverse-engineered so that the anonymity of the patient is protected. Such codes are obviously of enormous value for any data analysis of de-identified records.

13. The source of controversy is the language of the HIPAA rule itself. CFR §164.514 (c) states

> "Implementation Specifications: Re-identification. A covered entity may assign a code or other means of record identification to allow information de-identified under this section to be re-identified by the covered entity, provided that:
> (1) Derivation. The code or other means of record identification *is not derived from or related to information about the individual* and is not otherwise

capable of being translated so as to identify the individual [...]"

(emphasis added).

14. The crucial phrase is that any codes must "not [be] derived from or related to information about the individual[.]" Of course, encrypted or hashed linking codes *are* derived from identifying information, such as names, birth dates, etc. We can comment on this question only as statisticians, not as legal experts. Moreover, to our knowledge, the admissibility of such linking codes has never been tested in the courts, thus eliminating legal precedent as a source of guidance.

15. Speaking as scientists, the following interpretation is in our opinion the most reasonable and appropriate one. We take a code "derived from or related to information about the individual" to mean a code which contains identifying information in a form that is reasonably accessible to an intruder, perhaps with certain inside knowledge about the creation of the code. That is, a code which consists of identifying information or possibly contains a reduced or compressed record of identifying information is not admissible, if it is reasonable to assume that an intruder might be able to re-derive parts of that information. This is obvious from the HIPAA Rule which precludes such disclosures.

16. By contrast, a code created using strong cryptographic functions and and a sound cryptographic protocol will not offer an intruder a reasonable chance of extracting any part of the identifying information that was used to create it, provided any secret keys involved are kept securely by the covered entity or an appropriate third party business associate. Hence, even though technically the code is "derived" from identifying information in that the algorithm takes identifying information as input, we do not consider the code itself to be a "derived piece of information," because the code will be computationally indistinguishable from a random code to any intruder, and hence not offer any information about the individual.

17. To justify this interpretation we offer two arguments, one scientific and one legal.

(a) If we attempted to use the term of "a code derived from or related to information about an individual" in the more restrictive sense of "a code generated by an algorithm which uses identifying information as its input," we would have to acknowledge that, mathematically, the term "derived or related" code is ill-defined. If a *random* code was assigned to patient records, completely uncorrelated to any attributes of that patient or data available about that patient, it would still be possible, after the assignment, to construct an algorithm which would recreate these exact same random codes as a (suitably constructed) function of patient identifiers, say, names and birth dates. In fact, it would be very easy to do so, using only basic high school mathematics. While this would be a very artificial exercise, it shows that being "derived from or related to" information about a patient is *not* an intrinsic property associated with some types of codes but not with others. However, it does make sense to ask whether an intruder can be reasonably expected to be able to extract identifying information about a patient from an attached patient code. This is what we believe the HIPAA Rule asks for.

(b) We also refer to a discussion of the use of hashed message authentication codes (HMACs) in Vol. 67 of the Federal Register, August 14, 2002, p. 53233.

> "Several commenters who supported the creation of de-identified data for research [...] asked if a keyed hash message authentication code (HMAC) can be used as a re-identification code, even though it is derived from patient information, [...]
>
> Response: The HMAC does not meet the conditions for use as a re-identification code for de-identified information. [...] The covered entity may not share the key to the re-identification code with anyone, including the recipient of the data. [...] The HMAC methodology, however, may be used in the context of a limited data set [...]"

While this comment refers specifically to limited data sets, it seems to us that

HMACs would be similarly admissible on data sets used with a statistician's certification or in a safe harbor.

Therefore, we do believe that the use of hashed linking codes is fundamentally sound and in compliance with HIPAA.

3. THE ITRIALS SERVICE PLATFORM

3.1. Description

18. The creation of hashed linking keys is described as follows in the documentation we have received.

(a) Creation of a semi-unique "source string" from patient data. [...] This string is constructed so that a small number of collisions will occur. This is by design, and does not impact the iTrials execution model. The fields used for source string must have the following properties:

  (i) At least one of the strings must be of variable length, in order to prevent reverse-engineering of the hash by brute forcing constructed strings.

  (ii) The fields must be the same across multiple disparate data providers, in order to assure that different providers will construct the same source string.

(b) Passing this source string through an industry-standard approved hashing algorithm.

(c) Deleting the patient information used in the creation of the source string, and subsequently using the newly created hash as a medical history identifier.

[...] The small collision rate is a natural consequence of the fields selected for the creation of the hash:

(α) Patient last name (Variable length, punctuation dropped and forced to upper case for all providers)

(β) Patient Date of Birth

(γ) Patient Gender

These fields are concatenated and passed through an industry standard hashing algorithm (currently SHA-256); the resulting string is used as a semi-unique medical history identifier, and the source fields are adjusted (Patient last name is dropped, and date of birth is adjusted [...] Patient gender is preserved. Subsequently, a "whitening string" (effectively a secret key) is combined with (parts of) the hash value, and a second pass through the SHA-256 algorithm is performed.

19. The small number of collisions actually makes any attempt for re-identification even more difficult because the hashed patient linking codes are no longer completely unique to an individual. However, this will only play a minor role in the overall security architecture because it is quite possible that any such collisions could be recognized and removed on the basis of the variables contained in the de-identified file. For example, if a de-identified file retains information about ZIP codes and age, this would likely distinguish the colliding patient records.

3.2. Discussion

20. Collisions in the source string occur when distinct patients share the information aggregated in the source string. This happens essentially only if two patients share the same last name, gender, and date of birth, or if any minor discrepancies disappear after the standardization. These collisions lead to erroneous linking of distinct patients. This might impact the usefulness of the linking codes if collisions happened frequently, but it has no bearing on the technical functioning of the algorithm, and, if anything, it improves privacy and anonymity because re-identification is even harder if different patients share the same linking code. Thus collisions are of no HIPAA concern at all.

21. The iTrials platform offers various levels of de-identification. While the cryptographic hash process is always the same, there are differences in step (c) of Paragraph 18, Subsection 3.1 above - deletion of patient information. While names or exact birth dates can never be retained on a de-identified data set, it is possible, depending on circumstances, to retain indirectly identifying information such as years, sometimes even months of birth or gender. Such a determination, however, cannot be made in general, but depends on a statistical analysis of the data set which is intended for de-identification. Such an analysis will determine the risk of re-identification from the variables retained and only be able to certify that the data set can be considered de-identified if any risk of re-identification is found to be minimal. As discussed in Subsection 1.2, Paragraph 11, it is not within the scope of this document to provide such an analysis since no data sets have been reviewed. Instead, we restrict attention to the de-identification technology itself.

4. TECHNICAL DISCUSSION

4.1. Hash functions

22. A hash function is a basic cryptographic primitive with many uses in computer security. A b-bit hash function takes an arbitrary input $x \in \{0,1\}^*$ (we think of a computer file or data string as a sequence of bits - 0's and 1's) and maps it to a fixed length output of b bits:

$$H\colon \{0,1\}^* \longrightarrow \{0,1\}^b.$$

The basic idea is that such functions, to be cryptographically useful, must be easy to compute but hard to invert. Of course, since hash functions compress data of arbitrary length to a fixed length output, they are never invertible in a mathematical sense. Specifically, what we mean is that hash functions are required to have the following properties.

(a) *Pre-image resistance.* Let m be a uniform random variable on $\{0,\ldots,2^b-1\}$. Then, given a random value m, the expected amount of work $T(m)$ an attacker has to

perform to find an $x(m)$ with $H(x(m)) = m$ is $E[T] = c \cdot 2^b$ for some $c > 0$.

(b) *Second pre-image resistance.* Given a pair (x, m) with $H(x) = m$ the expected amount of work $T(x, m)$ for finding an $x' \neq x$ with $H(x') = m$ is $E[T] = c \cdot 2^b$ for some $c > 0$.

(c) *Collision resistance.* The expected amount of work T for an attacker to find a pair (x, x') with $x \neq x'$ and $H(x) = H(x')$ is $E[T] = c \cdot 2^{b/2}$ for some $c > 0$.

23. Collision resistance is the easiest property to violate due to the birthday paradox. Since the hash function takes 2^b distinct values, a set of random hash codes is likely to contain a collision once its size exceeds approximately $\sqrt{2^b} = 2^{b/2}$. It is easier to violate than the other properties since we do not require the collision to occur on any specific hash value $m = H(x) = H(x')$, but on arbitrary m.[3]

24. The collision resistance requirement forces us to make hash values at least twice as long as the required bit security. That is, for the hash function to be secure against an attacker who cannot perform 2^k operations we need the hash function to return an output of at least $b = 2k$ bits. This was one of the motivations for designing 256 bit hash functions which really offer only at most 128 bit security. This is still a very high level of security, comparable to the security of block ciphers using 128 bit keys. However, 128 bit hash functions such as MD5 offer only 64 bit security against collision attacks, and this may no longer be sufficient against powerful attackers. In this discussion, an "operation" is some reasonably defined unit of work - a computer clock cycle, the evaluation of H on a fixed length block of data, or some other well-defined unit of work.

25. In the HIPAA context, arguably, cryptographic hash functions exceed by far what is needed to protect codes against reverse engineering (see, however, Subsection 4.4).

[3]The classical birthday paradox states that for any random set of 23 people there is a greater than 50% chance that two of them share the same birthday within the year. This is not really a paradox, but is called that way because the number 23 is lower than what many would intuitively expect. Many cryptographic attacks are based on the fact that collisions on b bit random functions occur statistically after $b/2$ bit work, asymptotically for large b.

Finding collisions does not directly help an intruder in recovering the identity of patients whose information has been processed through a hash function. Similarly, if a pair (x, m) (or several such pairs) was (were) known to an intruder, and if the intruder was able to find $x' \neq x$ with $H(x') = m$, this, too, would not reveal any information to the intruder that he did not have before. (Of course, having such a pair in the first place may be a HIPAA violation since x is a patient identity.) Even a violation of the first property would not necessarily compromise the goals of HIPAA, because for most m, there are usually infinitely many pre-images x with $H(x) = m$, and hence finding any such pre-image would not necessarily help an intruder in finding the particular x corresponding to the patient's identity that was used to create m.

26. The above discussion shows that a good cryptographic hash function (satisfying the properties in Subsection 4.4) will make it impossible for an intruder to recover a patient identity x from the hashed code $m = H(x)$ alone by "inverting" the hash algorithm. If no information is available about x, and if x is completely random, it is virtually impossible to recover x, because to do so would violate property (a) in Paragraph 22. We therefore concur that the use of such hash functions is appropriate to achieve de-identification.

4.2. Standard hash functions

27. Having discussed the theoretical properties of hash functions and their applicability to HIPAA in Subsection 4.1, we now turn to existing hash functions and their ability to fulfill these properties. The most common hash functions are MD5, SHA-1, SHA-256, and RIPE-MD. Other hash functions exist that are constructed from symmetric block ciphers or one-way functions used in public key cryptography (such as squaring modulo Blum integers). MD5 creates 128 bit hashes, SHA-1 creates 160 bit values and SHA-256 has 256 bit hashes. RIPE-MD, too, has 160 bits in its standard version.

28. The SHA-family is derived from ideas developed by Ron Rivest in his MD5 algorithm. Certain design weaknesses discovered in MD5 have been improved in SHA-n. The SHA-family has been standardized and adopted as Federal Information Processing

Standard FIPS 180-2.

29. It is beyond the scope of this document to study the design of these functions is detail. We only give some relevant facts:

- SHA-256 is the most secure of the above mentioned algorithms because it uses the longest hash values and no fatal flaws in the design of SHA-256 are known. The most popular hash length, for example, in secure Internet traffic, is still only 160 bits. Thus 256 bits is considerably more secure, and even longer hash values are hardly even found in practical use. Therefore, we consider SHA-256 a highly secure, suitable choice of hash function.

- Some weaknesses are known to exist in both the MD and SHA families, but as we shall discuss below in Subsection 4.3, we do not believe them to be relevant in a HIPAA context. Moreover, it is not clear whether the underlying attacks even apply to the 256 bit version of the SHA family.

4.3. Recent cryptanalytic advances against hash functions

30. We have given a brief overview of the theoretical security requirements on hash functions in Subsection 4.1, and listed some actual examples in Subsection 4.2. However, it turns out that most of the hash functions in actual use (such as the ones mentioned in Subsection 4.2) do not fully meet these requirements. There has been substantial progress in recent years on finding collisions in hash functions, and even second pre-images do no longer seem to be as hard to find as previously thought, which is arguably a more severe flaw than collisions.

31. Among the early results against hash functions is Dobbertin's discovery, in 1996, of collisions in the compression function of MD5. Subsequently, full collisions have been found for MD5 by Wang, Feng, Lai, and Yu [3]. There are also computer implementations finding such collisions, see for example [1] for documentation. A particularly interesting

collision has been posted at [4], where two *meaningful* but different postscript files have been created, both having identical MD5 hashes. These results have provided sufficient grounds to declare MD5 obsolete as a cryptographic hash function. However, in the context of HIPAA de-identification, even these flaws would not necessarily compromise security since collisions or even *random* second pre-images would not necessarily be useful for unauthorized re-identification. Nevertheless, the prudent approach is to treat MD5 as a legacy algorithm and replace it as soon as possible even in HIPAA applications. Whenever full collision and pre-image resistance is needed, it should no longer be used at all.

32. SHA-0 and SHA-1, too, have been broken more recently. In a paper by Wang, Yin, and Yu [5], a method of finding collisions has been demonstrated using an expected 2^{69} SHA-1 operations, about 2000 times less than the expected 2^{80} operations. Still, while this justifies calling SHA-1 broken, the required work of 2^{69} SHA-1 operations is still prohibitively expensive, and it may take a little time before we see actual collisions published.

33. For SHA-256, the margin of security appears to be so high that, based on today's knowledge, collisions still seem to be far out of reach. Together with the fact that

- Collisions are the least serious of the three requirements on hash function security discussed in Subsection 4.1
- They are even less relevant to the type of application in a HIPAA de-identification engine
- The iTrials engine applies SHA-256 twice

we conclude that the use of SHA-256 in the iTrials engine, as documented, is safe and satisfies a very high security standard.

4.4. Attacks against linking codes

34. The discussion so far has focused exclusively on the security of the hash *algorithm*, that is, its ability to meet the requirements set out in Subsection 4.1. It is very important in cryptography to consider not only the security of algorithms but also the security of the protocol within which the algorithm is used. Concretely, we need to determine whether the overall *protocol* (i.e., the various steps performed to construct linking codes) allows an intruder to recover patient identities without breaking the hashing algorithm itself.

35. The most obvious attack possible is a "dictionary attack." Using "Kerkhoff's principle," we always have to assume that an attacker might know the algorithms and protocols used to create the linking code. Because the input (source string) to the de-identification engine is not a random string (contrary to the simple model assumption we have been making in places), an attacker would be able to construct (guess) plausible source strings. For example, an attacker might combine last names from a phone book with all possible birth days and genders. Let us say, there are $100 \times 365.25 = 36,525$ possible birth days (every fourth year is a leap year with 366 days), yielding $2 \times 36,525 = 73,050$ gender-birth day combinations. Combined with, say, 100,000 most common last names an attacker would most likely capture a significant section of the population with $7,305,000,000$ combinations of names, genders, and birth days. While this seems like a large number, the computation of < 10bn hash values is not impossible to an attacker with moderate resources. Restricting to a smaller set of common names, an attacker could even achieve a decent success rate at much reduced work.

36. This discussion shows that the use of a public, un-keyed hash function is not suitable in the HIPAA context because the source strings are not random, but a significant portion of them could be easily guessed and verified by an attacker. Consequently, an attacker could construct a look-up table of identities and matching linking codes. It is not admissible to assume that an attacker would not know the exact hash algorithm or how the source string is concatenated. Such information has proven very vulnerable

to reverse engineering in the past (see the reverse-engineering of the A5 algorithm, for example).

37. We, therefore, believe that it is essential that a *keyed* hash function be used which depends on a randomly chosen secret key of sufficient length so as to be secure against brute force search. At a minimum, such a key must be 128 bits long. In the context of the iTrials platform, the secret key is called "whitening." In cryptography, keyed hash functions are also called hashed message authentication codes (HMACs). We discuss them in Subsection 4.5 below.

4.5. Keyed hash functions

38. There are several ways of constructing keyed hash functions from standard hash functions. A good discussion is given in [2], §9.5. p.355. We briefly summarize them below. Let H be a standard hash function and k a randomly chosen secret key. We can define a keyed hash function $H_k(x)$ as follows. Let $L(x, a)$ be an operator that extracts the left a bits from a string x, $R(x, a)$ the operator extracting the right a bits from x, and let $\ell(x)$ denote the length of x in number of bits.

(a) *Secret prefix method.*

$$m = H(k||x)$$

(b) *Secret suffix method.*

$$m = H(x||k)$$

(c) *Envelope method with padding.*

$$m = H(k_1||0^r||x||k_2), \quad k = k_1||k_2$$

(d) *Alternative envelope method.*

$$m = H(k||0^{r_1}||H(k||0^{r_2}||x)).$$

(e) *Provisio envelope method.* This method proceeds in a number of steps as follows.

$$
\begin{aligned}
k_1 &= L(k, \ell(k)/2) && (1)\\
k_2 &= R(k, \ell(k)/2) && (2)\\
t_1 &= H(x) && (3)\\
t_2 &= L(t_1, \ell(t_1)/2) && (4)\\
t_3 &= k_1 || t_2 || 0^r || k_2 && (5)\\
t_4 &= H(t_3) && (6)\\
t_5 &= L(t_4, \ell(t_4)/2 + \alpha) && (7)
\end{aligned}
$$

The constant r in (5) may be selected so as to adjust the length of t_3, but has no particular security relevance. The constant α in (7) takes one of the values $-16, 0, 16$. The default value is $\alpha = 0$. The key has $\ell(k) = 8,192$ bits.

39. In a HIPAA context, all of these methods would be suitable. However, in a general cryptographic setting, the first two methods are not recommended because they allow an attacker to use a known plain text pair (x, m) to construct a new valid keyed hash for an extended input message x' constructed from x by suitable expansion in the back or front. The envelope methods prevent this attack.

40. While method (d) (alternative envelope method) is slightly superior to the simple envelope method (c) above, it might not be suitable in the iTrials application because the integration of the key needs to build upon an existing un-keyed hash value. Method (c) (envelope method with passing) can achieve this by a second call to H. That is, if x is an existing un-keyed hash, a keyed version can be produced by padding x with a sufficient number of 0's to make the length of $0^r||x$ a multiple of 512, and by enveloping the key and calculating $m = H(k||0^r||x||k)$. New codes can be created in a manner compatible with existing codes, which only need to be processed through a second round of hashing. In order to use Method (d) (alternative envelope method), *two* more calls to H would be needed, resulting in a total of three calls to H per code, which is likely

to be an unacceptably high workload for the computation of a single code. Method (e) is the method applied by Provisio. It uses the envelope technique and is very secure. at the same time, it allows the integration of existing linking codes $H(x)$ with only one additional call to H, resulting in a total of only two calls.

41. From our discussion of dictionary attacks, we conclude that it be necessary to introduce a secret key into the creation of hashed linking codes, using one of the methods (c) - (e) introduced in Paragraph 38 above. The iTrials platform satisfies this requirement. The key needs to be kept secure for as long as the method is in use and data processed by this method are available, and to establish a sufficient level of security, it must be chosen randomly and be at least 128 bits long. Moreover, procedural and technical safeguards are needed to protect the key and the engine hosting it, as we shall discuss below in Section 5. We have been assured that these precautions have been put in place in the iTrials engine.

5. PROCEDURAL SAFEGUARDS

42. Even though a certain level of security is inherent in the cryptographic design, it is always necessary to complement cryptography with and embed it into a framework of procedural safeguards that further reduce the risk of unauthorized re-identifications.

5.1. Keys

43. It is essential that keys must be protected very strongly. If the secret keys were ever compromised, the "dictionary" or brute-force search attacks discussed in Subsection 4.4 would become available to an attacker. The keys used by Provisio's de-identification tool are therefore held in escrow by a third party who is bound not to disclose them (unless it is legally required to do so, e.g., through a court order or to comply with law enforcement).

44. While it would be ideal to embed keys into tamper proof hardware so that they can-

not be compromised while residing at the site of a covered entity, this is often impossible or impractical from a business point of view. It is acceptable to embed the keys into the software, as is done in Provisio's implementation. To protect the keys, some techniques are used in the implementation to prevent them from being easily recovered (e.g., by a de-compiler).

45. We do not discuss the technical details of the key protections in place, but restrict to listing some general options that are available and suitable to accomplish that goal.

(a) The key could be encrypted itself, using a salt and a password to be entered by the user.[4]

(b) The software and keys should reside in protected areas of the IT system, so that only few users with administrative privileges can directly access the binaries and keys, even though a larger group of authorized users may be able to call and execute the program.

(c) The keys should be stored in protected memory during run time.

A number of technical resources exists that explore such secure programming techniques, see for example [6].

5.2. IT systems

46. The IT systems housing the iTrials platform have to always be physically secured against theft or tampering. Users require authorization to access the tool. Usage must be monitored and audited. Electronic protections of industry standard grade are to be in place against intrusion or tampering (firewalls, off-line systems, etc.). We consider industry standard solutions sufficient for most purposes.

[4]The term "salt" refers to a random string combined with the password entered by the user. It renders inefficient brute force searches of hashed passwords using precomputed tables which exploit the fact that passwords are of special form (alpha-numeric), rather than random binary strings.

47. The reason for requiring certain safeguards around the tool itself is again that reasonable efforts should be made to prevent a potential intruder from gaining access to the tool, which, again, would open up the possibility of identity disclosure by brute-force search.

5.3. Legal safeguards

48. Under Provisio's policies and procedures, the iTrials platform is only delivered to clients or customers with a legal agreement delimiting its authorized use. Appropriate legal safeguards are incorporated into any business agreements. In particular, the user of the iTrials engine

(a) must not use it for any purpose other than de-identification of medical data,

(b must not attempt to recover the key or de-compile the code,

(c) must not give access to the tool to any user who is not an employee or contractor of the data provider or who does not need to use the tool to perform the de-identification of medical data.

6. CONCLUSIONS

49. We conclude that the iTrials platform which creates hashed linking codes by two calls to the SHA-256 algorithm, applied to patient identifying information and a secret key of at least 128 bits ("whitening string"), implements a highly secure method of de-identification, suitable to meet the standard of CFR §164.514. The risk of unauthorized re-identification is, in our opinion, very small and acceptable under HIPAA, as long as the protocol is properly implemented and certain procedural safeguards are in place, as we have discussed in this document.

50. Some optional modifications have been discussed, and we concur that these would not have any adverse effect on the security of the technology if Provision chose to im-

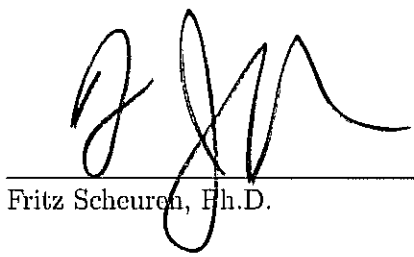
plement them. Since *random* collisions on a 256 bit string are virtually impossible, the security of the scheme could be further increased by discarding some of the output bits. This can be done in a number of ways:

(a) The final hash can be truncated (retain only some of the bits).

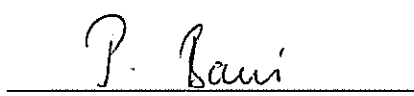
(b) Certain bits in the final output can be replaced with other information which is uncorrelated to the patient identity, but otherwise useful (e.g., system variables needed by the de-identification engine).

(c) The hash can be further compressed. For example, if we write the 256 bit string $H(m) = L||R$ as the concatenation of two halves, we could use $L \oplus R$ as linking code. Here $\oplus$ denotes bitwise exclusive or. This would be a 128 bit linking code, which would still offer the same functionality, increased performance (speed of record linkage, sorting and merging), and greater or equal security.

It is not necessary to implement any of these options from a security point of view, but they are optional modifications that would not require the engine to be re-certified.



Fritz Scheuren, Ph.D.



Patrick Baier, D.Phil.

Washington, D.C., 11/30/06

References

[1] Klima, V. Finding MD5 Collisions - a Toy For a Notebook.

[2] Menezes, A. and van Oorschot, P. Handbook of Applied Cryptography. CRC Press.

[3] Wang, X, Feng, D., Lai, X., Yu, H. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.

[4] `www.cits.rub.de/MD5Collisions/`

[5] Wang, X. Yin, Y.L., Yu, H. Finding Collisions in the Full SHA-1.

[6] Welschenbach, M. Cryptography in C and C++. Apress, 2001.