Sweeney L, Yoo J. De-anonymizing South Korean Resident Registration Numbers Shared in Prescription Data. *Technology Science*. 2015092901. September 29, 2015. http://techscience.org/a/2015092901

De-anonymizing South Korean Resident Registration Numbers Shared in Prescription Data

처방전 데이터의 주민등록번호 익명성 해제 연구

Latanya Sweeney, Ji Su Yoo

Highlights

- South Korea's national identifier, the Resident Registration Number (RRN) includes encoded demographic information and a checksum with a publicly-known pattern
- We conducted two de-anonymization experiments on 23,163 encrypted RRNs from prescription data of South Koreans
- We demonstrate the data's vulnerability to de-anonymization by revealing all 23,163 unencrypted RRNs in both experiments

Letter	Number
a	1
b	2
c	3
d	4
e	5
f	6
g	7
h	8
i	9
j	0

Odd-digit

Letter	Number
f	0
g	9
h	8
i	7
j	6
k	5
l	4
m	3
n	2
o	1

Even-digit

Coding table that replaced digits of South Korean national identifiers with letters in shared prescription data.

Abstract

When marketing and data analytic companies purchase medical data, sensitive medical facts about patients can leak if the information includes national identifiers. In some cases, patient information is at risk of exposure even when national identifiers are encrypted to provide privacy. We examined prescription data with encrypted national identifiers from South Korean decedents. Because the data did not include the patient's name or address and encrypted the patient's national identifier (Resident Registration Number, or RRN), the data was presumed to be anonymous and to resemble data shared with IMS Health, a large multinational corporation headquartered in the United States that collects these kinds of prescription data on millions of living South Koreans. However, weakly encrypted RRNs may be vulnerable to de-anonymization. They have demographics embedded within its digits with a publicly-known pattern, and like credit cards, a last digit that is a weighted sum of prior digits which allows for arithmetic inspection of the accuracy of re-identification. We conducted two experiments on the prescription data that demonstrates the vulnerability of its RRN encryption method. This work is timely because South Korea is debating a redesign of its national identifier, the United States is discussing a new universal patient identifier, and countries worldwide are struggling with breaches of personal information that rely on national identification systems.

Results summary: We conducted two de-anonymization experiments on 23,163 encrypted RRNs from South Korea. Each experiment independently revealed all 23,163 of the unencrypted RRNs and the results of each experiment validate the other since we matched the same RRNs to each of the same patients in both experiments. The first experiment relied on logical reasoning using the encoded design of the RRN. The second experiment computationally exhausted all possible decryption schemes to find those that satisfied the last digit calculation across all encrypted RRNs. Both experiments exposed the same letter-for-digit replacement scheme. Our findings demonstrate vulnerabilities with these encrypted identifiers and offer guidance on the use and design of national identifiers worldwide.

Abstract (Korean)

환자 의료정보에 개인식별 정보를 포함할 경우 의료정보 매매 과정에서 민감한 개인 정보가 유출될 수 있다. 심지어 주민등록번호와 같은 개인식별 정보를 암호화할 경우에도 환자 개인 정보가 마케팅 및 데이터 분석 업체에 노출될 수 있다. 본 연구는 이를 문제 삼아 대한민국 사망자의 처방데이터를 검토한다. 환자 주민등록번호를 암호화하고 이름 및 주소를 포함하지 않았기 때문에 동 데이터를 익명 데이터로 간주했으며 [IMS Health Korea] 라는 미국 대형 의료기업이 현재 한국인 수백만 명 대상으로 수집 중인 처방데이터와도 유사할 것으로 가정했다. 주민등록번호는 개인을 식별하는 정보를 포함하고 있다. 신용카드와 마찬가지로 주민등록번호의 마지막 숫자는 앞 숫자의 가중합산으로 생성된 것이라 주민등록번호의 유효 여부 계산

가능하다. 본 연구는 처방전 데이터를 이용해, 주민등록번호를 암호화 방식의 문제점을 보여준다. 현재 대한민국 정부는 주민등록번호 체계의 개편 논의 중이며, 미국은 유니버설환자번호를 모의 하는 등, 전세계 여러 나라가 개인 정보 유출 사건으로 피해를 입는 시점에서 본 연구가 시기 적절하다고 본다.

결과 요약: 본 연구는 암호화된 대한민국 주민등록번호 23,163 개를 대상으로 2 개의 익명성 해제 실험을 진행했다. 양 실험은 모두 원주민등록번호를 산출하고 동 주민등록번호를 해당 환자와 매치시키는 데에 성공했다. 첫 실험에는 주민등록번호의 암호화 체계를 바탕으로 한 논리적 추론을 이용했다. 두 번째 실험은 암호화된 주민등록번호의 마지막 숫자를 충족하는 모든 가능한 해독 방법을 적용하는 식으로 진행됐다. 양 실험은 모두 동일한 글자-대-숫자 대체 방법이 사용됐음을 밝혔다. 본 결과는 현재 사용 중인 식별 정보 암호화 방식의 약점을 드러내며 전세계 국가식별 정보 이용 및 구성 방법 관련 지침을 제공한다.

Introduction

South Korean residents must include their Resident Registration Number (RRN) every time they apply for a new credit card or process administrative paperwork for employment, banking, taxation, or social or medical services [1]. South Koreans also use RRNs extensively for online identification and payment. Having an individual's RRN makes it easier to impersonate the person and learn sensitive facts about him or her. Because South Korea is now a data-driven, highly networked society, unexpected problems involving RRNs arose.

In July 2011 there was a massive data breach at SK Communications, a South Korean conglomerate that owns Nate, a South Korean web portal, and Cyworld, a popular social networking platform [2]. Thirty-five million South Korean names, phone numbers, email addresses, and RRNs leaked, leaving many South Koreans at a high risk of identity theft and fraud. In August 2014, South Korean news sources reported that emails, passwords, and RRNs related to 104 million credit cards leaked from three major national credit card corporations [3].

South Korean medical software systems also store patient RRNs in order to track medical information, prescriptions, and insurance claims. Under South Korea's Personal Information Protection Act (PIPA), medical data is considered highly sensitive. No one can traffic personal data or identifiable health information on living patients without risking criminal penalties [4].

In February 2013, approximately 1,200 physicians and 900 private individuals filed a civil lawsuit against IMS Health, a large multinational corporation, for collecting the medical data

and RRNs of millions of living South Koreans [5]. IMS Health claims that it does not violate the privacy of South Koreans, in part because the RRNs are encrypted and cannot be decrypted by any reasonable means. Is this correct? Is it possible to decrypt these RRNs and associate actual RRNs with the medical information of patients to whom they belong?

These questions are timely. The South Korean government is considering restructuring their national identification system of RRNs [6]. The United States faces problems with leaks of identification numbers and personal information [7]. At the same time, some propose adoption of a new universal patient identifier [8] [9]. These countries are not alone. Table 1 lists 16 countries facing similar concerns [10]. The time is ripe to better understand the nature of the risks posed by national identifiers when anonymous data is widely shared.

Country	Name of the unique identifying number	Main uses of the identifying number
Belgium	INSZ NISS	INSZ NISS is a national person identifier used for various purposes, such as health care, social security, and tax.
Canada	Health Card Number	The provinces and territories assign a health card number that is a unique patient number for all publicly funded health care encounters. There is also a unique Social Insurance Number assigned nationally for tax and social security purposes. The Social Insurance Number is not used for health care.
Denmark	CPR N.R. (Central Person Register Number)	Used for "everything" in relation to national and local governments, including health care, banks, and other business identifications.
Finland	Personal Identity Code	The code is used in practically all data collections in public services, such as health care, social welfare services, education, justice, etc.
France	Numéro d'identification au répertoire (NIR)	Persons born in metropolitan France and overseas departments are registered on the national directory for the identification of natural persons (RNIPP) and assigned a registration number (NIR). The NIR is used by medical authorities for the issuance of a *carte vitale.* The NIR is also used for social security.
Israel	ID Number	The ID number is used for taxation, social security, education, health, licensing, banking, and other identified activities.
Italy	TS Number	The TS number contains both a health number and a tax file number. It covers nearly all the population. The TS number is managed through a publicly owned private company, Sogei, that could be considered a trusted third party.

Country	Name of the unique identifying number	Main uses of the identifying number
Malta	ID No	ID No is a unique identification number used throughout the country for all purposes, including electoral lists, taxation, and social security. It is based on the registration number at the Public Registry..
Norway	National Identification Number	The National Identification Number is used for taxation, social security, health records, banking, and other purposes.
Poland	PESEL	This number is used throughout the country for access to national health service care and benefits.
Portugal	Número de Utente do Serviço Nacional de Saûde	A PESEL number is assigned to all citizens at birth, permanent residents, temporary residents with stays of two months or longer, applicants for an identity card, and other persons where regulations require it.
Singapore	National Registration Identity Card Number (NRIC)	The NRIC is used for identification, government procedures, and some commercial transactions (e.g., opening a bank account)..
South Korea	Resident Registration Number	A Resident Registration Number (RRN) is assigned to each individual upon his/her birth and contains various information including birth date, gender and location of birth. RRN is used in virtually all aspects of life, including economic activities, for personal identification in various documents and communications in South Korea.
Sweden	Personnummer (Personal Identity Number)	Personnummer is the main identifier used for all official purposes in Sweden (taxation, social welfare, health care, living conditions, education, and so on).
United Kingdom	NHS Number. Scotland also has the Community Health Index (CHI) Number	Everyone registered with the National Health Service in England, Scotland, and Wales is issued a unique NHS number. The NHS number is not used for tax/social security purposes. In Scotland, the CHI system was set up for administrative purposes to track patients registering with GPs.
United States	Social Security Number	An SSN is issued to U.S. citizens, permanent residents, and temporary (working) residents primarily for taxation.

Table 1. Nationally assigned numbers that uniquely identify patients and the main uses of those numbers in different countries [10].

Background

The South Korean Resident Registration Number (RRN) is a 13-digit number. Each unique RRN explicitly encodes a person's demographic information. Birthdate, gender, and place of birth are all part of an RRN. The number follows the format YYMMDD – SBBCCAV [11]. Groupings of digits represent demographic encodings that can be deciphered with the information in Table 2. YYMMDD is the person's date of birth, where YY is the last two digits of the birth year, MM is the person's two-digit birth month (01 through 12), and DD is the two-digit day of the month (01 through 31). S is a single digit that encompasses gender and century of birth. BBCC is a code for the person's birthplace and community center. The birthplace is public information, as shown in Table 2, but the community center numbers (CC) are the only two digits whose encoded meanings are not publicly disclosed. Instead, the South Korean Ministry of Public Administration and Security keeps them private. A is a sequential number used to differentiate people of the same sex born on the same day in the same location. V is a check digit used to verify that the other digits are correct.

For example, 6405041024014 would be an RRN for a man born on May 4, 1964, in Seoul. The RRN 6405042024017 would be for a woman born the same day and in the same place. The RRN 6405042024022 would be for another female born on the same date and in the same place.

The verification code is the last digit. Its value depends on the preceding digits in the number itself. A verification digit is a quick way to compute whether a given RRN is correct. If someone enters a number and the checksum digit does not match, then the system can easily determine that the number is not correct. Checksum digits are most commonly used as the last digit in credit card numbers.

The checksum formula for RRNs appears in Table 2. If abcdef – ghijklm represent the first 12 digits of an RRN, then the last digit, V= (11 – [(2×a + 3×b + 4×c + 5×d + 6×e + 7×f + 8×g + 9×h + 2×i + 3×j + 4×k + 5×l) % 11] % 10), where % is the modulus or remainder operation. For the RRN 6405041024014, the last digit is 4. We can test whether this is a valid RRN by computing the last digit from the other digits and seeing if it is 4:

$$(11–[2\times6 + 3\times4 + 4\times0 + 5\times5 + 6\times0 + 7\times4 + 8\times1 + 9\times0 + 2\times2 + 3\times4 + 4\times0 + 5\times1] \% 11) \% 10$$

$$= (11 – [12 + 12 + 0 + 25 + 0 + 28 + 8 + 0 + 4 + 12 + 0 + 5] \% 11) \% 10$$

$$= (11 – [24 + 25 + 28 + 8 + 16 + 5] \% 11) \% 10$$

$$= (11 – [106 \% 11]) \% 10$$

$= (11 - 7) \% 10$

$= 4 \% 10$

$=4$

So 6405041024014 is a valid RRN.

Foreigners residing in South Korea receive an alien registration number (ARN) rather than a RRN. These numbers follow a different encoding pattern [12]. For simplicity, we limit this study to RRNs.

Birthdate (YYMMDD)	Sex & Century of Birth (S)	Birthplace & Community Center (BBCC)		Verification (AV)
YYMMDD format with no encoding	9: male born 1800 - 99 0: female born 1800 - 99 1: male 1900 - 99 2: female 1900 - 99 3: male 2000 - 99 4: female 2000 - 99 5: foreign male 1900 - 99 6: foreign female 1900 - 99 7: foreign male 2000 - 99 8: foreign female 2000 - 99	Seoul: 00-08 Busan: 09-12 Incheon: 13-15 Gyeonggi: 16-25 Gangwon: 26-34 N. Chuncheon: 35-39 Daejeon City: 40 Chuncheonnam: 41-47 Sejong City: 44, 96	Jeonbuk: 48-54 Jeollanam-do: 55-64 Gwangju: 65-66 Daegu: 67-70 Gyeongsangbuk: 71-80 Gyeongsangnam: 81-90 Ulsan: 85 Jeju: 91-95	A –12th digit to distinguish same sex and birthplace V – 13th verification digit: if RRN is *abcdef-ghijklm* then $V = (11 - [(2\times a+3\times b+4\times c+5\times d+6\times e+7\times f+8\times g+9\times h+2\times i+3\times j+4\times k+5\times l) \% 11]) \% 10$

Table 2. Encoding scheme of South Korea's 13-digit Resident Registration Number. The first six digits represent the birthdate in the format YYMMDD. The seventh digit represents a combined value for gender and century of birth. Digits 8 through 11 denote the birthplace and community center. Digit 12 sequentially increases to distinguish persons having the same birth date and place of birth. The last digit, 13, is a checksum calculation performed on the other digits.

Methods

The dataset we examined is a subset of a prescription record system of South Korean decedents from 2013. The file format appears consistent with descriptions of data provided to IMS Health from the same or a similar prescription record system [13]. Table 3 lists the fields of the dataset across two files. A row in one file links to a row in the other file by RRN. There are 23,163 unique values in the RRN field. Rather than being digits, however, the values in the RRNs fields are all lowercase letters, suggesting that the RRNs are encrypted using a substitution of letters for digits.

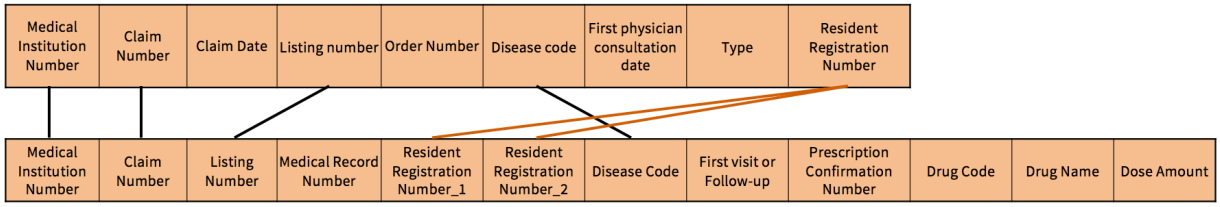

Table 3. Related fields in the disease file (top) and the prescription file (bottom). All the fields that comprise the disease file appear across the top row, from Medical Institution Number through Resident Registration Number. Some of the fields that comprise the prescription file appear along the bottom row, from Medical Institution Number through Dose Amount and beyond. Both files had labels that asserted they contained only the records of deceased people. In the diagram above, lines show which fields appear in both files. The encrypted RRN appears as 13 characters in the disease file, but in the prescription file, the encrypted RRN is split into two parts. The first part has the encoded birthdate (characters one through six), and the remainder, characters seven through 13, is the second part. There are 23,163 values in the RRN field of the disease file. All RRNs in the prescription file appear in the disease file, but some RRNs in the disease file have no corresponding entry in the prescription file. The additional fields in the prescription file beyond those listed in the bottom row are: frequency of dose (per day), form of medicine (capsule, tablet, etc.), duration of drug administration (in days), pharmaceutical manufacturer, hospital admission/outside, inpatient/outpatient, date of admission, reasons for hospital activity, dosage differentiation/type, 20% abnormality, and upper limit (purchase price or dosage).

For expediency, we refer to RRN encryptions appearing in the dataset as "encryptions." Because each encryption appears as a 13-character string of lowercase letters, we begin by parsing the letters by RRN digit position and calculating the frequency of each letter as it appears in each position. Specifically, we label the 13 character positions as digit1, digit2, … , digit13, from left to right, and report how many times we find each letter in each digit position. Table 4 summarizes our results.

Only the letters a though o appear in the encryptions, but not every letter appears in every position. For example, the first digit position (digit1) is the leftmost character position of the RRN and only the letters a through i appear there. The second digit position (digit2) only has the letters f through o. The table reveals an alternating pattern of letter assignments. Odd-numbered digit positions only hold characters a through j and even-numbered digits only hold characters f through o. Even though the letters f through j appear in both odd and even digit positions, the pattern encourages us to consider an odd-positioned assignment and a different even-positioned assignment for these letters. Therefore, the alphabet we consider for encryptions has 20 letters — {a, b, c, d, e, f1, g1, h1, i1, j1} for odd-numbered positions and {f2, g2, h2, i2, j2, k, l, m, n, o} for even-numbered positions. From this point forward, we assume that letters that appear in the odd-numbered positions use one set of letter-to-digit

assignments and those that appear in the even-numbered positions use another set of assignments.

RRN Format	Y	Y	M	M	D	D	S	B	B	C	C	A	V
Digit	**digit 1**	**digit 2**	**digit 3**	**digit 4**	**digit 5**	**digit 6**	**digit 7**	**digit 8**	**digit 9**	**digit 10**	**digit 11**	**digit 12**	**digit 13**
a	2402	0	5433	0	7898	0	7867	0	1484	0	2301	0	4279
b	400	0	0	0	7482	0	10174	0	2797	0	2762	0	2047
c	2066	0	0	0	817	0	2727	0	2549	0	2400	0	2090
d	3029	0	0	0	0	0	2394	0	3004	0	2519	0	2130
e	3878	0	0	0	0	0	1	0	4780	0	2496	0	2145
f	3549	2624	0	1865	0	2911	0	4847	1849	2497	2174	0	2121
g	2336	2496	0	1858	0	1947	0	231	1165	2596	2172	1	2072
h	1510	2278	0	1889	0	2171	0	434	1673	3174	2076	3	2066
i	1255	2280	0	1775	0	2337	0	332	1436	1876	2336	5	2146
j	2738	1954	17730	1651	6966	2011	0	1998	2426	1590	1927	21	2067
k	0	2017	0	1849	0	3184	0	2769	0	2083	0	92	0
l	0	1945	0	1955	0	1945	0	2940	0	2064	0	240	0
m	0	2093	0	2189	0	2146	0	2827	0	2620	0	931	0
n	0	2805	0	4038	0	2097	0	3262	0	2692	0	4008	0
o	0	2671	0	4094	0	2414	0	3523	0	1971	0	17862	0
Letters Present	a-j	f-o	a/j	f-o	j,a,b,c	f-o	a-e	f-o	a-j	f-o	a-j	g-o	a-j

Table 4. The frequency of each character, a through o, at each digit position, 1 through 13, in the encrypted RRNs. Digit 1 is the leftmost character position of the RRN and only letters a through j appear in the leftmost position. Digit 2 is the character that is second from the leftmost position and has letters f through o. Digits 3, 5, 7, 9, 11 and 13 (rightmost digit) only have letters a through j while digits 4, 6, 8, 10, and 12 have letters f through o. The bottommost row lists the letters that appear for each digit position. The topmost row repeats the RRN encoding scheme described in Table 2.

Table 4 also suggests that each encryption is independent of any other field of information appearing in the data. Therefore, we only need the encrypted RRNs for our study. We strip away all medical and other fields from consideration. All further references to our study data or dataset in this writing are to the list of 23,163 encrypted RRNs alone unless otherwise specified or obvious from context.

Our method involves two independent experiments, each capable of identifying the letter-for-digit assignments used in the encryptions. One experiment uses human-guided logical reasoning. The other experiment uses a computer to exhaustively search through the space of all possible solutions. Below are descriptions of each experiment.

Logical Reasoning Experiment

In our logical reasoning experiment, we systematically inspect occurrences of values and make inferences about which letters must correspond to which digits based on knowledge of RRN encodings (Table 2). We assume the subjects are an older random sample of the general population.

For example, a two-digit representation for a month has to start with a 0 or a 1; these are the only possible values in the 3rd digit of an RRN. Replacing these digits with letters will mean that one of the letters replaces 0 and the other replaces 1. Further, if the first digit is 1, then the second digit can only be a 0, 1, or 2 for the months October, November and December, respectively. We can then determine which of the letters must be 0 and which must be 1 by observing how many different letter combinations appear in the 3rd and 4th positions. We continue with similar reasoning on other character positions.

Overall, our inference experiment has three phases. In the first phase, we perform a series of logical deductions about values that must appear alone and in combination to determine which letters must represent which digits. We use the date, gender, and sequence values appearing in digits 3, 6, 7 and 12. In the second phase, we use the checksum formula to compute values for those letters that we did not learn in the first phase. In the third phase, we verify the correctness of the letter-to-digit assignments we deduced by verifying the checksum for all the encryptions.

Finally, we realize that data from real-world practice will include alien registration numbers, not just RRNs, and may include a few errors. We end this experiment with an investigation of any encryptions that do not satisfy our derived digit-for-letter assignments. We clean the data accordingly so that what remains are qualified RRNs for the next experiment.

Automated Search Experiment

In our automated search experiment, we use a computer to exhaust all possible letter-to-digit assignments that can account for the encryptions. The most straightforward approach is to test which substitutions made for every possible digit-letter combination satisfies the checksum constraint across all the encryptions. Unfortunately, there are 13 digit positions and 20 possible letters, which gives 1320 letter-to-digit assignments to test. Trying all these combinations blindly, even on today's high-powered computers, is impractical. Therefore, we only try those combinations that can account for the letter combinations that actually appear in the encryptions.

Here is an example of our basic approach. In these examples, the letter j appears in odd-numbered positions and the letter f appears in even-numbered positions so we will just write f and j. Consider the encryption afjojodoeodoj. Its letters are a, d, e, f, j, and o. Substituting all possible digits for these letters gives 106 or a million possible combinations. However, not all of these possibilities have a value of j that matches the checksum calculation:

(11–((2×a + 3×f + 4×j + 5×o+ 6×j + 7×o + 8×d + 9×o + 2×e + 3×o + 4×d + 5×o) % 11)) % 10

= (11–((2×a + 3×f + 10×j + 29×o+ 12×d + 2×e) % 11)) % 10

To demonstrate, we substitute 0 for all the letters. In this case, j=0, and the checksum calculation is: (11–((2×0 + 3×0 + 10×0 + 29×0+ 12×0 + 2×0) % 11)) % 10, which is 0. Therefore, replacing all letters with 0 is a possible solution. Next, we substitute 1 for j and 0 for all the other letters. Clearly, this cannot be a solution because the checksum calculation will still be 0. In fact, none of the combinations that substitute any non-zero digit for j and substitutes 0 for all the other letters will satisfy the checksum constraint. We can express this the other way. There is only one value for j that satisfies the checksum calculation for any combination of digits for the other letters. Therefore, rather than one million possible combinations for this one encryption, there are only 100,000.

We then ask, of these 100,000 possible encryptions, how many will also satisfy the checksum calculation for all the other encryptions in the dataset? Consider the encryption aoaoaodododoj. The checksum digit is also j, but the letters are arranged differently, so the checksum calculation is different:

(11–((2×a + 3×o + 4×a + 5×o+ 6×a+ 7×o + 8×d + 9×o + 2×d + 3×o + 4×d + 5×o) % 11)) % 10

= (11–((12×a + 32×o+ 14×d) % 11)) % 10

One of our previously possible solutions substituted 0 for all letters. That combination satisfies this checksum calculation also, because j is 0 and the checksum calculation is 0. However, many of the other 100,000 possibilities will not longer satisfy both checksum calculations. For example, substituting 6 for a and 0 for the other letters gives the following checksum calculations. For encryption afjojodoeodoj, the checksum digit j is 0 and the calculation is:

(11–((2×6 + 3×0 + 10×0 + 29×o+ 12×0 + 2×0) % 11)) % 10

= (11–(12 % 11)) % 10

= (11-1) % 10

= 10 % 10

= 0

Therefore, the checksum digit and the checksum calculation match for encryption afjojodoeodoj. For encryption aoaoaodododoj, however, the checksum digit j is 0 and the calculation is:

$$(11-((12\times6 + 32\times0+ 14\times0) \% 11)) \% 10$$

$$= (11-(72 \% 11)) \% 10$$

$$=(11 -6) \% 10$$

$$=5$$

The checksum digit and the checksum calculation do not match for encryption afjojodoeodoj. Therefore, we can no longer consider the combination that substitutes 6 for a and 0 for the other letters as a solution. In fact, of the 100,000 solutions possible for the first encryption, only 10,738 of them satisfy both checksum calculations. After processing these two encryptions, the number of possibilities reduces to 10,738.

We then continue processing encryptions, where all possible combinations of substitutions of digits for letters in each new encryption processed must satisfy the checksum calculation for its encryption as well as the checksum calculations for all prior encryptions processed. Eventually the process converges to one or few possibilities for the entire dataset. It collapses to none if there is no single combination of digits to substitute for letters that will satisfy the checksum calculations for all encryptions.

To efficiently express our approach, we begin with a few definitions. The "alphabet" was described earlier as 20 letters, {a, b, c, d, e, f1, g1, h1, i1, j1} for odd-numbered positions and {f2, g2, h2, i2, j2, k, l, m, n, o} for even-numbered positions. We write a pair of strings to list letters appearing in the odd and even character positions, respectively. For example, (adj, fo) describes the letters a, d, and j appearing in the odd positions and the letters f and o appearing in the even positions. So, (abcdefghij, fghijklmno) expresses the entire 20-letter alphabet.

We call encryptions that use the same odd and even letters, but in different permutations, "same letter permutations" of each other. For example, the encryptions aoafafdfcfaod, eoaobfafahcoc, and cfaoahbfjfeoc are same letter permutations of each other because they only use (acd, fo). Same letter permutations allow us to test the validity of digit-to-letter assignments, because no matter the permutation, the checksum requirement should remain valid across all same letter permutations if those digit-to-letter assignments are correct.

A "cipher" is our way of expressing assignments of digits to none, some or all of the 20 letters in the alphabet. Any letter can assume any digit 0-9. We use the value -1 to specify that a letter is not bound to a digit; we say it is unbounded. We write ciphers as rows in a table to denote them as a set of ciphers. Table 5 shows examples of three sets of ciphers. The first set has cipher A, a cipher in which none of the letters have digits. This is the starting or empty cipher.

The set of ciphers B and C in Table 5 assign digits to the same letters (acdej, fkmno) and demonstrate an interim state of the search. A cipher's digit-for-letter substitutions satisfy all same letter permutations if the checksums are valid. Cipher B and Cipher C bind the same numbers to letters and they are both valid for the same set of encryptions examined so far in the search.

Cipher D in Table 5 assigns a digit to each letter. We call it a full cipher. If checksums are valid for all encryptions using a full cipher's digit-for-letter substitution, then the cipher is a solution.

	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
A	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
B	1	-1	3	4	5	-1	-1	-1	-1	0	0	-1	-1	-1	-1	5	-1	-3	2	1
C	9	-1	9	9	9	-1	-1	-1	-1	9	1	-1	-1	-1	-1	1	-1	1	1	1

	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
D	1	2	3	4	5	6	7	8	9	0	0	9	8	7	6	5	4	3	2	1

Table 5. Three sets of ciphers stored as tables. Each row is a cipher that assigns digits to letters in the 20-letter alphabet of the encryptions. Letters a through j1 appear in odd positions. Letters f2 through o appear in even positions. A letter can assume any digit, 0-9. The value -1 means that the letter is not bounded. The first table having row (A) is a cipher that binds no letters. The second table has ciphers (B) and (C) which assign digits to (acdej, fkmno). The last table has cipher (D), which binds a digit to each letter in the alphabet.

Here are some general statements about our use of ciphers. A set of ciphers may include ciphers over different letters from each other. The letters covered by the set is the union of all the letters covered by its ciphers. A cipher does not apply to an encryption that uses letters unbounded by the cipher. A set of ciphers is valid if all same letter permutations of the union of its ciphers produce valid checksums from all applicable ciphers.

The computer searches for a cipher that assigns a digit to each letter in the alphabet and whose assignments satisfy the checksums of the encryptions. Figure 1 provides the algorithm for our approach. The computer starts with the empty cipher (Step 1). It then finds an encryption in the dataset that has the least number of letters unbounded by the ciphers in the table (Step 2). The first time through the program, this will be the encryption in the dataset having the least number of letters. The computer then generates all ciphers (i.e., all digit-to-letter possibilities) that yield valid checksums for the encryption (Step 3).

At this point, the number of ciphers in the table will remain the same or increase. The number of ciphers in the table will remain the same if more letters become bound by existing ciphers.

The number of ciphers increases by including all additional ciphers that can also account for the encryption.

The program tests the ciphers against all same letter permutations and removes any cipher that no longer satisfies the checksum requirement (Step 4 and Step 5). The number of ciphers in the table will remain the same or decrease.

Steps 2 through 5 repeat until either the cipher table is empty (because no solution was found), or the cipher table has only full ciphers (because ciphers assign digits to each letter). Any full cipher in the table at the end of the program is a solution. If the program concludes with one and only one full cipher that accounts for all the digits, it is the only mathematical solution possible. All other possibilities were eliminated.

Step 1.	The table has the empty cipher. No letters are bound.
Loop	
Step 2.	Find an encryption that has the least number of letters unbounded by the cipher table.
Step 3.	Add any new ciphers to the table capable of validating this encryption.
Step 4.	Find all same letter permutations for the cipher table.
Step 5.	Remove any ciphers from the table that are not valid for the same letter permutations.
Repeat until the cipher table is empty or has only full ciphers	

Figure 1. Algorithm for exhaustive search of digit-to-letter assignments based on the encryptions appearing in the dataset.

Results

Logical Reasoning Experiment

We begin with the first 6 digits of an RRN, which stores birthdate in the format YYMMDD (see Table 2). Months in this format are values 01 through 12. In Table 4, only two possible characters appear for the first digit of the month (the digit3 position); these are the letters a and j. This j is in an odd-numbered position (digit3), this j is j1 in the alphabet. So, either a=0 and j1=1, or a=1 and j1=0.

To disambiguate, we examine how values for digit4 pairs with values for digit3. In an RRN, when digit3 is a 0, digit4 can be 1 through 9, and when digit3 is 1, digit4 can only be 0 through 2 (for months 10, 11 and 12). How do these appear in the dataset?

If j1=0, then digit4 must exhibit 9 different letters. Alternatively, if j1=1, then digit4 must exhibit 3 different letters. Table 6 shows the counts. When digit3 is a j, nine different letters appear in digit4. When digit3 is an a, three different letters appear. Therefore, j1=0 and a=1.

Letter	if digit3= 'j', count when digit4= 'letter'	if digit3= 'a', count when digit4='letter'
a	0	0
b	0	0
c	0	0
d	0	0
e	0	0
f	0	1865
g	1858	0
h	1889	0
i	1775	0
j	1651	0
k	1849	0
l	1955	0
m	2189	0
n	2249	1789
o	2315	1779

Table 6. Counts of pairings of digit3 and digit4 in the dataset. When digit3 is the letter j, 9 different letters appear in digit4 (left). When digit3 is the letter a, 3 different letters appear in digit4 (right).

Now we look at digit5, which stores the leftmost digit of a two-digit day. Because days in this format are numbers 01 through 31, there are only four possible numbers for digit5 (0, 1, 2 and 3). In Table 4, only 4 letters (j1, a, b, and c) appear in digit5. So {j1, a, b, c} must be {0, 1, 2, 3}, but which letter is which digit?

Earlier we determined that j1=0 and a=1. So, either b=2 and c=3, or b=3 and c=2. To disambiguate, we focus on the fact that some months have 30 days and other months have 31 days. If c=3 in digit5, then digit6 will be restricted to two values (0 or 1). Conversely, if c=2 in digit5, then digit6 will be restricted to ten values (0 through 9). Table 7 shows how digit5 pairs with digit6. When digit5 is c, digit6 has only with 2 possible letters. When digit5 is b, digit6 has 10 different letters. Therefore, b=2 and c=3. Table 8 has a summary of the four letter-to-digit assignments deduced so far.

Letter	if digit5= 'c' , count when digit6='letter'	if digit5='b', count when digit6="letter"
a	0	0

Letter	if digit5= 'c' , count when digit6='letter'	if digit5='b', count when digit6="letter"
b	0	0
c	0	0
d	0	0
e	0	0
f	643	1194
g	0	568
h	0	753
i	0	749
j	0	643
k	0	927
l	0	673
m	0	670
n	0	669
o	174	636

Table 7. Counts of pairings of digit5 and digit6 in the dataset. When digit5 is the letter c, 2 different letters appear in digit6 (left). When digit5 is the letter b, 10 different letters appear in digit6 (right).

Letter	Number
a	1
b	2
c	3
d	
e	
f (of f1)	
g (or g1)	
h (or h1)	
i (or i1)	
j (or j1)	0

Odd-digit

Letter	Number
f (or f2)	
g (or g2)	
h (or h2)	
i (or i2)	
j (or j2)	
k	
l	
m	
n	
o	

Even-digit

Table 8. Coding table deductions after analysis of the first digit in 2-digit representations of months MM and days DD as stored in positions digit3 and digit5. We write the values f, g, h, i, and j as f1, g1, h1, i1 and j1 when they occur in odd-numbered digit positions and as f2, g2, h2, i2, and j2 when they occur in even-numbered digit positions to distinguish them.

Seven months have a 31st day. When DD is 31, digit5 is 3 and digit6 is 1. We already know that c=3 (see Table 8). We examine values appearing in the encryptions for digit6 when digit5 is a c

and find digit6 has only two possible letters: f2, which appears 643 times, and o, which appears 174 times. These numbers of occurrences suggest that f2 is 0 and o is 1 because every month that has a day 31 also has a day 30, but not the other way around. There are more day 30s than there are day 31s. Therefore, f2=0 and o=1.

In the MMDD format, DD is 31 only when MM is 01, 03, 05, 07, 08, 10, or 12. Two of these months, 10 and 12, begin with a 1. These values of MMDD are: 1031 and 1231. We already know that a=1 and c=3 for the odd-numbered digit positions, and f2=0 and o=1 for the even-numbered digit positions. We examine values for digit4 appearing in the encryptions when digit3 is an a, digit5 is a c and digit6 is an o. The only nonzero occurrences are: afco, which appears 23 times; and, anco, which appears 20 times. We already know that afco or a(f2)co is 1031. So, anco is 1231. Therefore n=2. Table 9 has a summary of the seven letter-to-digit assignments deduced so far.

Letter	Number
a	1
b	2
c	3
d	
e	
f (of f1)	
g (or g1)	
h (or h1)	
i (or i1)	
j (or j1)	0

Odd-digit

Letter	Number
f (or f2)	0
g (or g2)	
h (or h2)	
i (or i2)	
j (or j2)	
k	
l	
m	
n	2
o	1

Even-digit

Table 9. Coding table deductions after analysis of digits for the MMDD positions. We write the values f, g, h, i, and j as f1, g1, h1, i1 and j1 when they occur in odd-numbered digit positions and as f2, g2, h2, i2, and j2 when they occur in even-numbered digit positions to distinguish them.

Digit7 stores the gender and century information. Among decedents, we expect most people to have been born in the 1900's and fewer in the 2000's. We examine the frequencies of values for digit7 in Table 3. Letters a and b have the most occurrences at 7867 and 10174, respectively. Letters c and d have fewer at 2727 and 2394, respectively. These frequencies suggest that letters a and b are for genders of people born in the 1900's and letters c and d are for people born in the 2000's. We already know that a=1, b=2, and c=3. Therefore, d=4.

Digit8 stores the sequential number assigned to people who have the same birth demographics. Most people will not have a duplicate, so they will have a 1 in digit8. Of those having duplicates, having one additional duplicate is more likely than having two additional

duplicates, which is more likely than having three additional duplicates, and so on. We expect that the smaller the number, the greater the number of occurrences. We examine the frequencies of values for digit8 in Table 3. There is a strict ordering from the letter o having 17862 occurrences, n having 4008 occurrences, m having 931, l having 240, k having 92, j having 21, i having 5, h having 3, and g having one occurrence. We already know that o=1 and n=2. Therefore, m=3, l=4, k=5, j2=6, i2=7, h2=8, and g2=9. Table 10 has a summary of the letter-to-digit assignments deduced so far. We have learned all the even-numbered digit assignments.

Letter	Number
a	1
b	2
c	3
d	4
e	
f (of f1)	
g (or g1)	
h (or h1)	
i (or i1)	
j (or j1)	0

Odd-digit

Letter	Number
f (or f2)	0
g (or g2)	9
h (or h2)	8
i (or i2)	7
j (or j2)	6
k	5
l	4
m	3
n	2
o	1

Even-digit

Table 10. Coding table deductions after analysis of digits for the MMDD and S positions. We write the values f, g, h, i, and j as f1, g1, h1, i1 and j1 when they occur in odd-numbered digit positions and as f2, g2, h2, i2, and j2 when they occur in even-numbered digit positions to distinguish them.

We compute the remainder of the digits for letters e through i in the odd-numbered positions using the checksum formula for digit13. We search the encryptions to find an encryption that has the letter e for digit13 and letters in digits1 through digit12 whose values we already know (see Table 10). We find the encryption: ckanakanckdne. We enter the known values for the letters and calculate the checksum for letter e:

$$e = 11-((2\times c + 3\times k + 4\times a + 5\times n + 6\times a + 7\times k + 8\times a + 9\times n + 2\times c + 3\times k + 4\times d + 5\times n) \% 11)$$

$$= 11-((2\times 3 + 3\times 5 + 4\times 1 + 5\times 2 + 6\times 1 + 7\times 5 + 8\times 1 + 9\times 2 + 2\times 3 + 3\times 5 + 4\times 4 + 5\times 2) \% 11)$$

$$= 11-((6+ 15 + 4 + 10 + 6 + 35 + 8 + 18 + 6 + 15 + 16 + 10) \% 11)$$

$$= 11-(149 \% 11)$$

$= 11-6$

$= 5$

Therefore, e=5. We repeat the process and find encryption dgafagajdgaof having a checksum letter of f. We already know values for all its other letters. (see Table 10).

$$f = 11-((2\times d + 3\times g + 4\times a + 5\times f + 6\times a + 7\times g + 8\times a + 9\times j + 2\times d + 3\times g + 4\times a + 5\times o) \% 11)$$

$= 11-((2\times 4 + 3\times 9 + 4\times 1 + 5\times 0 + 6\times 1 + 7\times 9 + 8\times 1 + 9\times 6 + 2\times 4 + 3\times 9 + 4\times 1 + 5\times 1) \% 11)$

$= 11-((8 + 27 + 4 + 0 + 6 + 63 + 8 + 54 + 8 + 27 + 4 + 5) \% 11)$

$= 11-(214 \% 11)$

$= 11-5$

$= 6$

Therefore, f1=6. We repeat the process and find encryption bnjkjkbkbhjng having a checksum letter of g, encryption cfaoajakcjcoh having a checksum letter of h, and encryption dmaobiambmdoi having a checksum letter of i. Other than the checksum digit, we know the values for all the other letters in these encryptions so we compute the checksum digits, using the same process shown above. We learn g1=7, h1=8, and i1=9. We have now inferred all the letter-for-digit assignments. Table 11 shows a complete list.

Letter	Number
a	1
b	2
c	3
d	4
e	5
f (of f1)	6
g (or g1)	7
h (or h1)	8
i (or i1)	9
j (or j1)	0

Odd-digit

Letter	Number
f (or f2)	0
g (or g2)	9
h (or h2)	8
i (or i2)	7
j (or j2)	6
k	5
l	4
m	3
n	2
o	1

Even-digit

Table 11. Coding table for the RRN encryptions. We write the values f, g, h, i, and j as f1, g1, h1, i1 and j1 when they occur in odd-numbered digit positions and as f2, g2, h2, i2, and j2 when they occur in even-numbered digit positions to distinguish them.

To test whether our derived assignments are correct, we substitute numbers for letters for each of the encryptions in the dataset and, for each encryption, we compare its value for digit13 with the checksum we calculate from its other digits. The checksum matches in 23,161 of 23,163 (or 99.991 percent) of the encryptions.

The two encryptions for which the checksums do not work are fojhjneiffahe and aoaoajdldldld. The first, fojhjneiffahe, decodes to 6108025760185. According to Table 2, the 5 in digit7 identifies it as a registration number for a foreigner (ARN). As noted earlier, these numbers follow a different encoding. The second encryption, aoaoajdldldld, decodes to 1111164444444. While this could be a female born on November 16, 2011, there are no other encryptions having birthplace 4444 in the dataset and the checksum digit should be 1 instead of 4. This encryption seems to be an error. We remove these two encryptions from the dataset and continue to the next experiment with 23,161 encryptions in the dataset.

Automated Search Experiment

We conduct a second experiment to learn the coding of the RRN encryptions. In this experiment, we make no inference about dates or distributions of values. Instead, we view each encryption as a checksum equation over the alphabet. The sufficiently large number of encryptions in the dataset will provide enough equations for the computer to learn the values assigned to each letter. Figure 1 lists the steps of the algorithm.

At the start of the program, the cipher table is empty. No letters are bound to any digits. The encryption with the fewest letters is aoaoaodododoj. It generates 10,000 ciphers in the table for letters (adj, o). There are no same letter permutations of this encryption in the dataset. As examples, the cipher that substitutes 6 for a, 7 for d, 9 for j1 and 3 for o is one of these 10,000 ciphers. The cipher that substitutes 1 for a, 4 for d, 0 for j1 and 0 for o is another of the 10,000 ciphers.

On the second iteration, the encryption with the next fewest unbounded letters is afjojodoeodoj. This adds 738 ciphers over the letters (adej, fo) after validating the checksums of 2 same letter permutations. As an example, the cipher that substitutes 6 for a, 7 for d, 9 for j1 and 3 for o in the prior iteration, changed to additionally substitute 2 for e, 0 for f2.

On the third iteration, the encryption with the fewest unbounded letters is aoafafdfcfaod. This adds one cipher to the table, and the letters covered by the table are (acdej, fo).

On the fourth iteration, the encryption with the fewest unbounded letters is dfanafafcoaoa. The number of possible ciphers dramatically drops from 10,739 to 11. Only 11 ciphers can satisfy all possible substitutions of digits for letters in the checksum calculations imposed by the encryptions aoaoaodododoj, afjojodoeodoj, aoafafdfcfaod, and dfanafafcoaoa.

The algorithm in Figure 1 executes for 16 iterations. Table 12 describes the state of the computer at the conclusion of each cycle. Starting with iteration 4, the computer maintains

11 ciphers that become increasingly more refined by binding more letters to digits within the ciphers. Table 13 shows the evolution of two of these ciphers from iteration 4 through iteration 12.

When execution ends after iteration 16, the table has the 10 ciphers shown in Table 14. Each cipher describes digit-to-letter substitutions that satisfy the checksum equations for all encryptions.

As a final step, we test these digit-to-letter assignments by substituting the prescribed numbers in each encryption in the dataset and then calculate the checksum digit for agreement with its value in the 13th position. The checksum is correct in all 23,161 encryptions (or 100 percent). The perfect score results from our excluding the two non-RRNs in this experiment that we included in the prior experiment.

While all 10 ciphers in Table 14 mathematically satisfy the checksum constraint imposed on each encryption, ciphers #2 through #10 do not account for all possible digits that can actually occur in RRNs. Only cipher #1 has an assignment for each digit; therefore, it the only viable solution. It assigns: 1 through 9 to the letters a through i in odd-numbered positions; 0 to j in odd-numbered positions and to f in even numbered positions; and, 9 through 1 to the letters g through o in even-numbered positions. This solution is identical to the results in Table 11 from the prior experiment.

Iteration	Encryption with fewest unbounded letters	Number of ciphers in table	Letters bounded by ciphers in table	Number of same letter permutations processed
1	aoaoaododododoj	10000	(adj, o)	1
2	afjojodoeodoj	10738	(adej, fo)	2
3	aoafafdfcfaod	10739	(acdej, fo)	3
4	dfanafafcoaoa	11	(acdej, fno)	10
5	jnjnandnamanj	11	(acdej, fmno)	26
6	emjnjnakenjnj	11	(acdej, fkmno)	68
7	ekafahafafaoa	11	(acdej, fhkmno)	206
8	ekafbfaoefaoa	11	(abcdej, fhkmno)	602
9	ifafbkafbfbna	11	(abcdeij, fhkmno)	1025
10	dmafafaodgama	11	(abcdeij, fghkmno)	1819
11	dfafafaodgfoa	11	(abcdefij, fghkmno)	2886
12	gfjhanahjhjnj	11	(abcdefgij, fghkmno)	4025
13	dojoaoaogigog	10	(abcdefgij, fghikmno)	6465
14	enanjoaleneoj	10	(abcdefgij, fghiklmno)	10829
15	ghaoajahboaog	10	(abcdefgij, fghijklmno)	16682
16	eoaoaiamhmjoa	10	(abcdefghij, fghijklmno)	23161

Table 12. States of the algorithm (Figure 1) as it iterates 16 times to compute a list of letter-to-digit assignments that satisfy the checksums of all the encryptions in the dataset. Execution begins with an empty cipher table, grows to as many as 10739 ciphers in iteration 3, and ends with 10 ciphers. Each cipher in the final set explains all the encryptions in the dataset (see Figure 2). Each pair of strings shows the letters bounded by the ciphers in the table at that

iteration. The left string lists letters in odd-numbered positions and the right string lists letters in even-numbered positions.

	Evolution of Cipher #1 Letter to Digit Assignments																			
Iteration	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
4	1	-1	3	4	5	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	2	1
5	1	-1	3	4	5	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	3	2	1
6	1	-1	3	4	5	-1	-1	-1	-1	0	0	-1	-1	-1	-1	5	-1	3	2	1
7	1	-1	3	4	5	-1	-1	-1	-1	0	0	-1	8	-1	-1	5	-1	3	2	1
8	1	2	3	4	5	-1	-1	-1	-1	0	0	-1	8	-1	-1	5	-1	3	2	1
9	1	2	3	4	5	-1	-1	-1	9	0	0	-1	8	-1	-1	5	-1	3	2	1
10	1	2	3	4	5	-1	-1	-1	9	0	0	9	8	-1	-1	5	-1	3	2	1
11	1	2	3	4	5	6	-1	-1	9	0	0	9	8	-1	-1	5	-1	3	2	1
12	1	2	3	4	5	6	7	-1	9	0	0	9	8	-1	-1	5	-1	3	2	1

	Evolution of Cipher #2 Letter to Digit Assignments																			
Iteration	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
4	9	-1	9	9	9	-1	-1	-1	-1	9	1	-1	-1	-1	-1	-1	-1	-1	1	1
5	9	-1	9	9	9	-1	-1	-1	-1	9	1	-1	-1	-1	-1	-1	-1	1	1	1
6	9	-1	9	9	9	-1	-1	-1	-1	9	1	-1	-1	-1	-1	1	-1	1	1	1
7	9	-1	9	9	9	-1	-1	-1	-1	9	1	-1	1	-1	-1	1	-1	1	1	1
8	9	9	9	9	9	-1	-1	-1	-1	9	1	-1	1	-1	-1	1	-1	1	1	1
9	9	9	9	9	9	-1	-1	-1	9	9	1	-1	1	-1	-1	1	-1	1	1	1
10	9	9	9	9	9	-1	-1	-1	9	9	1	1	1	-1	-1	1	-1	1	1	1
11	9	9	9	9	9	9	-1	-1	9	9	1	1	1	-1	-1	1	-1	1	1	1
12	9	9	9	9	9	9	9	-1	9	9	1	1	1	-1	-1	1	-1	1	1	1

Table 13. Evolutionary states of the first 2 ciphers in the cipher table from iteration 4 through iteration 12. During these iterations, the computer manages 11 ciphers, two of which appear above as top and bottom tables. A letter that is not bound has the value -1. Bounded letters have the assigned digit. During these iterations, the computer modifies these ciphers to satisfy the checksums imposed by more encryptions and it does so by increasingly binding more letters without increasing the number of ciphers.

	Letter to Digit Assignment																			
Cipher	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
#1	1	2	3	4	5	6	7	8	9	0	0	9	8	7	6	5	4	3	2	1
#2	9	9	9	9	9	9	9	9	9	9	1	1	1	1	1	1	1	1	1	1
#3	7	7	7	7	7	7	7	7	7	7	2	2	2	2	2	2	2	2	2	2
#4	5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3	3	3
#5	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
#6	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4
#7	1	1	1	1	1	1	1	1	1	1	5	5	5	5	5	5	5	5	5	5

Cipher	Letter to Digit Assignment																			
	a	b	c	d	e	f1	g1	h1	i1	j1	f2	g2	h2	i2	j2	k	l	m	n	o
#8	8	8	8	8	8	8	8	8	8	8	7	7	7	7	7	7	7	7	7	7
#9	6	6	6	6	6	6	6	6	6	6	8	8	8	8	8	8	8	8	8	8
#10	4	4	4	4	4	4	4	4	4	4	9	9	9	9	9	9	9	9	9	9

Table 14. Cipher table when algorithm (Figure 1) concludes. It has 10 ciphers. The top row identifies the letters to which the cipher assigns digits. Cipher #1 assigns 1 through 9 to the letters a through i in odd-numbered positions of an encryption or RRN. It assigns 0 to j in odd-numbered positions and to f in even numbered positions. Finally, it assigns 9 through 1 to the letters g through o in even numbered positions. This is the only cipher in the table that can be a solution because none of the other ciphers (#2 through #10) include all the digits.

Discussion

Both experiments concluded with the same letter-to-digit assignments. The first experiment derived the letter-for-digit substitutions using human guided logical reasoning with computer assistance from a common spreadsheet program (Excel). The second experiment used a custom program to accomplish the task through automated means. Together, these experiments demonstrate that the encrypted RRNs are vulnerable to almost any adversary.

The encrypted RRNs were associated with patient prescription and medical information as described in Table 3. Using decedent information, our experiments demonstrate how others could associate an actual RRN for a living patient with his sensitive medical information. IMS Health was named in a lawsuit as having purchased this kind of prescription information [5]. If IMS Health did receive these kinds of data, then our study exposes the real-world realities of imperfect anonymization. Further research is necessary to propose alternatives to lawsuits.

Our experiments exposed other general problems too, including the choice of encryption, the short length (13 digits) of RRNs, the existence of a checksum digit in the RRN, and the encoding of demographics in RRNs. These vulnerabilities offer opportunities for further research.

As a cryptosystem, our experiments exposed a simple substitution cipher. Each digit is replaced with a lowercase letter using different substitutions for even and odd digit positions. This replacement scheme is too easy to learn. Stronger forms of encryption are available.

Of course, encryption always has a key for decryption. Does the existence of a decryption key make encryption inappropriate for sharing patient health information that is supposed to be anonymous? As our experiments revealed, with a key, it is just as easy to encrypt as it is to decrypt. If the recipient of the data should be unable to decrypt the encrypted RRNs, then a better approach is one that is easy to encrypt but hard to decrypt. This suggests the use of a strong hash function such as MD5 [14] and not encryption.

Even with strong hashing, vulnerabilities can persist. National identifiers usually have few digits (such as 10 or 13) so that people can remember them. However, the fewer the digits, the easier it is for a computer to exhaust all hashed or encrypted possibilities. For example, Sweeney showed that even when 10-digit U.S. Social Security numbers (SSNs) were hashed using a strong hash function, the original SSNs could still be learned because the hash algorithm was shared widely. In a matter of seconds, her computer hashed all SSNs from 000-000-0000 to 999-999-9999. She stored the computations in a lookup table along with their originating SSNs. To de-anonymize shared SSNs, she would simply look up the hashed value in her table to identify the SSN [15]. This limitation inspires further research to be done on alternative approaches to national identifiers.

Another vulnerability is the checksum digit in RRNs because it imposes a mathematical relationship on the digits. Credit card numbers also have a checksum digit, so our experiments encourage further studies on practices that require encrypted or redacted credit card numbers.

Finally, the encoding of demographics into RRNs poses vulnerabilities for data sharing because that same information may appear in other fields. Most data studies include date of birth, gender and residential postal code. If these fields had been in the prescription data, then half the digits of the RRN would have been explicitly visible. This would have made the RRNs even easier to decrypt.

Changing or replacing a national identifier is difficult. Experts estimate the cost for South Korea to do so to be about 3100 to 4000 hundred million Korean won (around 2.6 billion USD) [16]. On the other hand, not updating national identifiers poses economic risks for the systems and individuals that use them.

What, if anything, might our findings suggest for the design of new national identifier systems? Randomization would have made our effort much more difficult, if not impossible, even using the same encryption scheme. A randomized identifier would have no encoding of date of birth, gender, or any personal information, no checksum digit, and no inferences about the person assigned the number. There are, of course, other methods to consider, so the best design for national identifier systems seems an open question ripe for future work.

References

1. Jo D. Study on Resident Registation Number Collection Alternative System by Using Personal Information. Chung-Ang University. 2013. http://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=4ab51723b1cbadfbffe0bdc3ef48d419

2. Lee D. North America technology. Millions Hit in South Korean Hack. BBC News. Web. 31 Dec. 2014. http://www.bbc.co.uk/news/technology-14323787

3. Kong K. South Korea Credit-Card Issuers Suspended. Wall Street Journal February 16, 2014. Web. August 8, 2015. http://www.wsj.com/news/articles/SB10001424052702304899704579386250111246222.

4. Laws Concerning the Resident Registration Number. Privacy Information Protection Portal. Ministry of Government Administration and Home Affairs. http://privacy.go.kr/a3sc/law/waa/intSummary.do

5. 2014 Form 10K for IMS Health Holdings. IMS Health Ltd. 2014. p.14, 27. http://ir.imshealth.com/files/51a1975c-604b-4a89-b19b-6d36fa633e19_v001_g8065f.pdf.

6. Rah D. Discussing the RRN system overhaul" (Ministry of Security and Public Administration hearing, 6 suggestions). September 29, 2014. Kookmin News. Web. September 2, 2015. http://news.kmib.co.kr/article/view.asp?arcid=0922799212&code=11131100&sid1=cen

7. Millions More Americans Hit by Government Personnel Data Hack. Reuters. Web. 9 July 2015. Web. September 10, 2015. http://www.reuters.com/article/2015/07/09/us-cybersecurity-usa-idUSKCN0PJ2M420150709

8. Identity Crisis? Approaches to Patient Identification in a National Health Information Network. RAND Corporation. Web. September 9, 2015. http://www.rand.org/pubs/research_briefs/RB9393/index1.html

9. Peel D. Should Every Patient Have a Unique ID Number for All Medical Records? Wall Street Journal January 23, 2012. Web. September 9, 2015. http://www.wsj.com/articles/SB10001424052970204124204577154661814932978

10. OECD Health Policy Studies Strengthening Health Information Infrastructure for Health Care Quality Governance Good Practices, New Opportunities and Data Privacy Protection Challenges: Good Practices, New Opportunities and Data Privacy Protection Challenges. OECD Publishing, 2013. Print. p. 52 http://www.oecd.org/els/health-systems/Strengthening-Health-Information-Infrastructure_Preliminary-version_2April2013.pdf

11. Seoul High Court. July 18, 2002. Sentence 2001, Section 16776 Decision cited in Lee S. The Constitutional Evaluation on Resident Registration System: Focuses on the Resident Registration Number and Fingerprinting. Hanyang University. Web. September 9, 2015. http://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=5851f031376756e1ffe0bdc3ef48d419

12. Korea Federation of Banks. Finance Documentation. 2006. http://www.kfb.or.kr/cms.html?S=GAAAB

13. IMS LifeLink Information Assets Longitudinal Prescription Data. IMS Health. 2012. https://www.imshealth.com/deployedfiles/ims/Global/Content/Solutions/Healthcare%20Analytics%20and%20Services/Healthcare%20Outcomes/IMSLifelink-LongitudinalRx.pdf

14. Rivest R. The MD5 Message-Digest Algorithm. Memo. MIT and RSA. April 1992. Web September 9, 2015. https://www.ietf.org/rfc/rfc1321.txt

15. Sweeney L. Risk assessments of personal identification technologies for domestic violence homeless shelters. Carnegie Mellon University, School of Computer Science. Technical Report CMU-ISRI-05-133. Pittsburgh. November 2005. http://dataprivacylab.org/projects/homeless/paper1.pdf

16. Kang E. Influential Introduction of Newly Issued Resident Registration Number. Digital Times. 29 Sept. 2014. Web. September 9, 2015. http://www.dt.co.kr/contents.html?article_no=2014092902100351800001

Authors

Latanya Sweeney is Professor of Government and Technology in Residence at Harvard University, Director of the Data Privacy Lab at Harvard, Editor-in-Chief of Technology Science, and was formerly the Chief Technology Officer of the U.S. Federal Trade Commission. She earned her PhD in computer science from the Massachusetts Institute of Technology and her undergraduate degree from Harvard. More information about Dr. Sweeney is available at her website at latanyasweeney.org.

Ji Su Yoo graduated from Harvard College with a B.A. in Social Studies. As an undergraduate, she spent two summers doing field research in South Korea and completed an honors thesis on Korean Unification. Currently, she is interested in the intersection between government and technology – especially how their interaction can impact privacy and data analysis in health care. Ji Su hopes to attend graduate school and to continue investigating impactful research projects.

The authors gratefully thank Nuri Kim for his help with Korean translations.

Editor: James Waldo

Citation

Sweeney L, Yoo J. De-anonymizing South Korean Resident Registration Numbers Shared in Prescription Data. *Technology Science*. 2015092901. September 29, 2015. http://techscience.org/a/2015092901

Data

Under review for data sharing classification. Data release available October 19.