I.DNA

The Ultimate Form of Security for your Personal Identity

Daniel Garcia, Logan Gatenby, and Dalton Scharff
University of Pittsburgh

Abstract

I.DNA is a secure personal identification system designed to reduce and eliminate identity theft. By utilizing the human genome and the safest of card reading technologies, I.DNA binds your identification number directly to who you are. You're not just an identification number, your identification number is you.

Reasoning

In September 2017, Equifax revealed that a security breach in their database left upwards of 140 million users' data at risk. Included in this data was user's names, addresses, and social security numbers.

Social security numbers have evolved to become one of the most basic forms of identification for United States citizens. While originally intended only to keep track of an individual's Old-Age, Survivors, and Disability Insurance, these 9 insecurely-generated numbers have come to be required for a user to obtain nearly any type of account. Obtaining a driver's license, creating a bank account, getting a job, and even paying taxes require a social security number. This exposes your personal identification number to many hazards, including data leaks as was the case with Equifax.

A social security number is composed of three sections. The first three digits, since 1973, have been based on the requestor ZIP code. The next two are group numbers which increment with a system based on the amount of numbers within a ZIP code. The final four are serial numbers, which index straight from 0000 to 9999. Therefore, even when a corporation only asks for the final four digits of your social security number, it's just as unsafe as revealing the whole number, as the first 5 digits can be found on a lookup table.

An identification number with so much power should have an equally powerful security system backing it. The current method of national identification does not follow these security principles. Instead, it relies human memory and trust, hardly an infallible system.

Process

Genome

The human genome is composed of approximately 3 billion base pairs (i.e., ‘A’, ‘C’, ‘G’, or ‘T’). This, straight out of the DNA sequencer, is 200 gigabytes of data, which is far too large to be quickly processed by any of today’s technology. However, the amount of genome mutations (i.e., the base pairs that vary among humans) totals about 3 million base pairs, or 125 megabytes, straight from the DNA sequencer. This is a more manageable amount of data, but we can do better.

Hashing

In order to shrink the list of genome mutations (currently about 125 megabytes) down, SHA256 can be used. Secure Hash Algorithm 256 can take a message of any length (e.g., our list of genome mutations), and converts it to 256 bits, or just 32 bytes. At this size, every living human could have their hashed genome mutation stored in less than 250 gigabytes, smaller than the average laptop hard drive.

$$\frac{\text{32 bytes per genome mutation * 7.6 billion people on Earth}}{\text{\textasciitilde 1 billion bytes per gigabyte}} = \text{243.2 gigabytes}$$

Dual Factor Authentication

EMV, developed by Europay, MasterCard, and Visa, is the protocol backing the technology that goes into smart payment cards (i.e., chip cards). Instead of account information being stored in the chip, replace this with an individual’s genome mutation hash. This can then be used at terminals similar to the ones used for payments in order to verify a person’s identification, and may have expanded uses down the road.

Technological Explanation

Genome

One of the most secure forms of security is to utilize what you “are” (i.e., a piece of your physical person), as this cannot be forgotten or lost. However, since there is no algorithm or definition to define what makes you *you*, researches have come close by using techniques to quantify an individual's uniqueness. Known as biometrics, or literally assigning numbers to someone's biology, sounds futuristic, but is actually in widespread use today.

Fingerprint scanners, often included on most modern cell phones, or facial recognition, included in Windows 10's "Windows Hello" software as well as the iPhone X, are each forms of biometrics. However, these become faulty when these features change. Faces change over time and fingerprints can be modified. Even without actual modification, these forms of biometrics have faults because they are based on a point mapping system. This means, for example, that instead of comparing two images for exact replicas, software looks for numerous points of interest on an original scan (the one you made during initial setup) and tries to match that with points of interest on the current scan (while you are trying to currently unlock your device). While this does not seem inherently bad, consider all of the changes that are made to your face throughout the day. Maybe a new blemish appears on your chin or you decided to grow your hair out to cover your ears. These are potential differences between the original and current scan. If companies developed their software to only let users in on perfect matches, it would be nearly impossible to regain entry into your device, as your look is always changing. These "false negatives" would lead to very unhappy users, meaning that software developers leave room for differences (i.e., if in a 1000 point mapping where 990 points match, the software would allow the user to gain entry). This creates the opportunity for "false positives" though, as perhaps a different individual who happens to look nearly identical may be granted entry as well.

The human genome mutation list has a much lower chance of false positives. Since an individual's DNA does not change over time, any two samples of DNA collected from an individual should be expected to match 100% of the time. Therefore, the only opportunity of false positives could be from identical twins, which can be accounted for.

Hashing

A hash function is a function that maps any length of input to a fixed length code. In the case of SHA256, this is 256 bits or 32 bytes. One beneficial aspect of hashing functions are that they are one-way. Therefore, given a hashing algorithm and a pre-hashed message, it should require brute-force (i.e., in the worst case, testing all possible inputs) in order to find the original input. For example, given a hashing function H and a hashed output z, it should be infeasible to find the input message x without first hashing all the possible inputs. For a 32 bit number, such as the one that SHA256 generates, this translates into 1.2E77 different inputs, a large number even for a computer. If someone were to obtain someone else's hashed genome mutation list, they would not be able to reverse engineer the function to find out to whom it belongs without running through all possible inputs.

Another important aspect of hashing is that it should be infeasible to find to messages that, when each are hashed, return the same result. When two hashed, differing inputs create the same output, this is called a collision. Good hashing functions have various ways of preventing this, which is crucial to this application, as a collisions would create another source for false positives.

Hashing also changes the output significantly upon a slight change in input. Adding or modifying a single character in the input string will change the output in an unpredictable fashion. This means that most siblings (except for, again, identical twins) will have vastly differing hashes of their genome mutation list.

SHA256 was developed by the National Security Administration and released for public use in 2001. Not only has it been adopted for security by industries worldwide, but it also is required for government usage on secured information, which is an added testament to its security. The message space (i.e., the size of the output) is significantly large as there should not be any collision conflicts, and if there were, the algorithm can handle it. Finally, changing the input to the function slightly significantly changes the output.

EMV

Communicating data between devices can be unsafe. However, what good is data if it only ever sits in memory? Consider the magnetic strip on a credit card. That strip holds a digital representation of the same numbers that are physically embossed or printed on the card itself. These numbers are accessible by any card reader, no matter if the reader belongs to a trustworthy bank or a malicious scammer trying to sell your credit card number on the black market.

EMV has worked to change this, as it is a technical protocol designed to protect against credit card skimmers, or “man-in-the-middle” attacks. With a magnetic strip, once a credit card number was compromised, it was gone for good. If left unreported, the man-in-the-middle can partake in infinitely many transactions without being found out. The technology behind EMV forgoes this by creating a unique number with each transaction and verifying that with a trusted remote server, where the hash is actually stored. If someone were able to pull the number from a single transaction at an EMV terminal, they wouldn’t receive the card owner’s genome mutation list or hash, but instead a one-time-use number based on, but not including the genome mutation hash, that would not be valid for any further transactions.

While the United States has only recently adopted the EMV system for their payment cards, European countries have been using it and have trusted it for years. EMV terminals are quite commonplace and consumers are becoming accustomed to using them, leaving this a feasible and easily integratable solution to the problem of security for personal identification.

Obtaining the Initial Card and Hash

Ideally, the card and hash would be created at birth, just as a birth certificate is. However, it would be possible to implement in anyone's life at any point.
To receive a hash and card:

1. Have a DNA sample collected at a certified facility
    a. The sample will be sent to a lab to be sequenced
    b. The lab will limit the sequence to just mutations
    c. The genome mutation list will be run through the SHA256 hashing algorithm
2. If the recipient is an adult, a photo must be taken to be printed on the card
    a. Otherwise, the recipient will need to obtain a photo version of the card at age 18

Resolving Disputes

If a malicious individual were somehow able to reverse engineer the algorithm that creates the EMV transaction numbers in an attempt to steal someone's identity, obtaining proof that you are who you claim to be is simple.

To resolve a dispute:

1. Obtain DNA samples from both individuals involved in the dispute at a certified facility
2. Sequence each individual's DNA
3. Limit the sequence to just the mutations
4. Run the genome mutation list through the SHA256 hashing algorithm
5. Whichever output from step 4 matches the disputed hash is the real possessor of that identity

Potential Issues

Identical Twins

Identical twins have the potential to share the exact same genotype, which would produce two of the exact same hashes for these individuals. To account for this, each genotype mutation list could be appended with a number. The first born would get a genotype mutation list appended with a 1, the second with a 2, and so on. Due to one of the properties of a hash function, each twin would receive an entirely unique hash.

Data Leaks

After a hash is created from an individual's genome mutation list, that list should be securely discarded and not stored. The hashes must be stored, though, and the security of the database that holds the hash should be to the strictest of standards. However, in the unlikely event that a leak were to occur, it would be possible to regenerate a new hash from the current (leaked) hash appended with the time of creation. This time of creation would then be stored in the new, secure database along with the original hash.

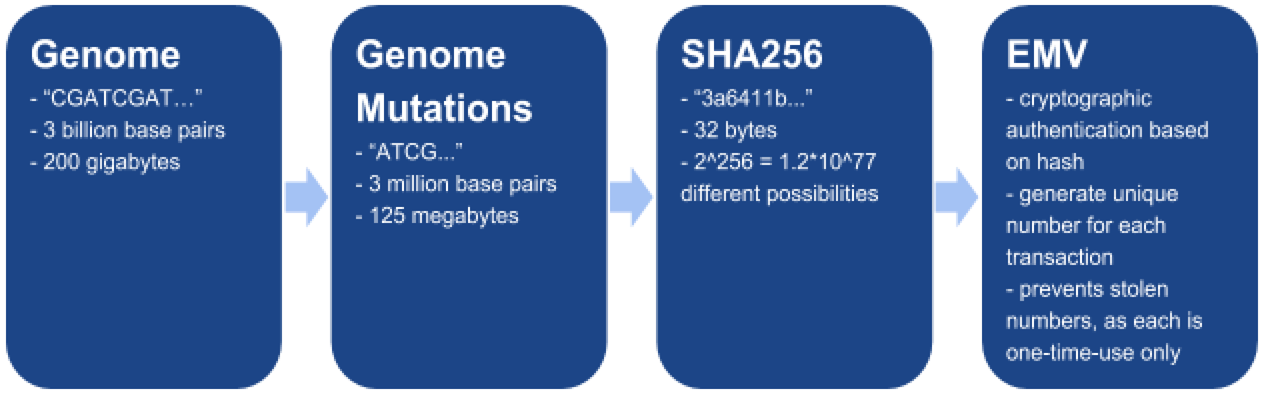
For example, say John Doe's genome mutation list is simply "ACGAT".
SHA256(ACGAT) = ac3650babbb3dc726d7370f418b373bc278e104aa418cd62ecab7f5fa59109f2

Hash	**First Name**	**Last Name**	**...**
ac3650babbb3dc726d7370f418b373bc278e104aa418cd62ecab7f5fa59109f2	John	Doe	...

After a breach, we could rehash the compromised hash appended with a random integer.
SHA256(ac3650babbb3dc726d7370f418b373bc278e104aa418cd62ecab7f5fa59109f2 || 0945714411) = 14668689a6cd8506e070b2b36e188373be079a665344df0796dec660ad14fbe0

Hash	**Random Number**	**First Name**	**Last Name**	**...**
14668689a6cd8506e070b2b36e188373be079a665344 df0796dec660ad14fbe0	0945714411	John	Doe	...

Data Flow


Genome
- "CGATCGAT..."
- 3 billion base pairs
- 200 gigabytes
Genome Mutations
- "ATCG..."
- 3 million base pairs
- 125 megabytes
SHA256
- "3a6411b..."
- 32 bytes
- 2^256 = 1.2*10^77 different possibilities
EMV
- cryptographic authentication based on hash
- generate unique number for each transaction
- prevents stolen numbers, as each is one-time-use only

Additional Resources

EMV

https://www.level2kernel.com/emv-guide.html

Equifax Leak

https://www.forbes.com/sites/winniesun/2017/10/02/what-you-should-do-now-after-the-equifax-security-leak/#78928e332123

Genome

https://medium.com/precision-medicine/how-big-is-the-human-genome-e90caa3409b0

Hashing

https://en.wikipedia.org/wiki/SHA-2
https://github.com/CS1653/lecture-slides/blob/master/lecture06-hash.pdf

Social Security Number

https://en.wikipedia.org/wiki/Social_Security_number

Twin Differentiation

http://www.fsigenetics.com/pb/assets/raw/Health%20Advance/journals/fsigen/FSIGEN_monozygotic_twins.pdf