De-identification Isn't Anonymization, and Neither Is a Weak Hash

Wed, Sep 30, 6:00 PM EDT · https://scottslab.io/posts/deidentification-anonymization-data-obfuscation

TL;DR

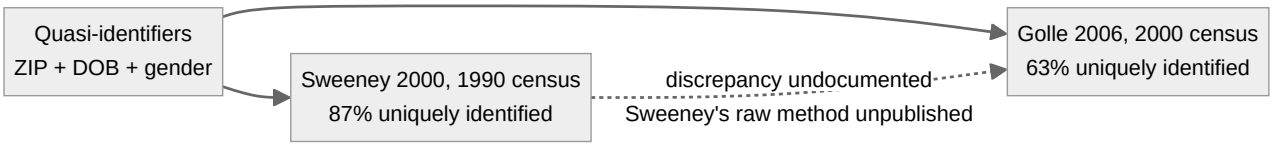
Stripping the obvious identifiers off a dataset feels like anonymization and isn't; a small set of quasi-identifiers like zip code, date of birth, and gender can re-identify a large share of a population on their own, though exactly how large is a genuinely contested number. HIPAA's Safe Harbor method lists 18 specific elements to remove, which is a compliance bar, not a re-identification guarantee. And of the four common obfuscation techniques, hashing is the one people trust the most and should trust the least, because it's only as strong as how guessable the input was to begin with.

De-identification is a checklist. Anonymization is an outcome.

De-identification means you removed a defined list of fields. Anonymization means nobody can reasonably reconstruct who the record belongs to. Those are not the same claim, and treating a de-identified dataset as anonymous is where a lot of privacy incidents actually start, not with a broken database, with a dataset that technically passed the checklist and still identified people.

The reason is quasi-identifiers: fields that aren't names or Social Security numbers on their own, but narrow a population down to almost nobody once combined. The canonical demonstration is Latanya Sweeney's 2000 Carnegie Mellon paper, *Simple Demographics Often Identify People Uniquely*, using 1990 Census data: 87% of the U.S. population, 216 million of 248 million people, had a combination of 5-digit ZIP code, gender, and full date of birth unique to them alone. Three fields nobody flags as sensitive on a form, and for most people that combination was as identifying as a name.

That 87% is the number that gets quoted everywhere, and it's worth being precise about what happened to it since. A 2006 re-analysis by Philippe Golle, using 2000 Census data and a fully documented methodology, found a meaningfully lower figure: 63% uniquely identifiable by the same three fields, and 61% when he re-ran his method against 1990-era data. Golle couldn't fully explain the gap, since Sweeney's original data collection wasn't published in enough detail to replicate exactly, and he speculated it may partly trace to how the 1990 Census tabulated ZIP codes versus counties. The honest summary isn't "87%," and it isn't "63%" as a settled replacement. Two independently documented studies, a decade and a census apart, agree the number is large and disagree sharply on how large. Citing this without the caveat is citing a number, not the finding.



De-identification is a checklist. Anonymization is an outcome.

What Safe Harbor actually asks for

HIPAA's Privacy Rule offers two routes to de-identifying protected health information: Expert Determination, where a qualified statistician certifies the re-identification risk is very small, and Safe Harbor, a fixed checklist. Safe Harbor requires removing 18 specific categories of identifier: names; geographic subdivisions smaller than a state (a narrow exception applies to the first three digits of a ZIP code in large-enough areas); all elements of dates more granular than year, including birth date, and ages over 89; phone and fax numbers; email addresses; Social Security numbers; medical record and health plan beneficiary numbers; account numbers; certificate or license numbers; vehicle and device identifiers; web URLs; IP addresses; biometric identifiers; full-face photographs; and a catch-all for any other unique identifying number, characteristic, or code, provided the covered entity has no actual knowledge the remaining data could still identify someone.

Notice what's on that list and what isn't. Safe Harbor strips date of birth down to year alone, and it doesn't ask you to generalize ZIP code beyond that narrow three-digit exception, and it doesn't touch gender at all: close to the exact three fields the Sweeney/Golle work is about. A dataset can satisfy Safe Harbor's literal checklist and still carry a quasi-identifier combination with real re-identification risk, which is precisely why the rule includes the actual-knowledge clause: passing the list doesn't excuse ignoring a known way back to the individual.

Four techniques, one weak link

Hashing runs a value through a one-way function so the output can't be reversed to the input directly. That's true, and also close to irrelevant when the set of possible inputs is small and known. I've written elsewhere about hashing passwords, where the candidate space is effectively unbounded, exactly what makes salting so effective there. A Social Security number has nine digits: a billion possible values, fully enumerable. A date of birth has maybe 40,000 realistic values across a working lifetime. An attacker doesn't crack a hashed SSN, they precompute the hash of every possible SSN once, ahead of time, and it's a lookup, not a computation. That precomputed table is a rainbow table, exactly as effective against a hashed SSN as it's ineffective against a well-chosen password.

Salting is the direct fix for table reuse, the same mechanism as password salting: mix in a random, unique value before hashing so a precomputed table stops matching. It only helps if the field is

actually salted, and a lot of hashed-SSN implementations skip the step because "we already hashed it" reads as done. But salting a keyspace this small isn't the win it is for a password: it kills table reuse, yet recomputing one salt's billion-value SSN space with a fast hash is still cheap per record, so a targeted, per-record attack isn't meaningfully slowed the way it is against a real password.

Tokenization sidesteps the whole hashing question by not deriving the replacement value from the original data at all. A random, unrelated identifier gets generated and stored in a secure lookup table that maps token back to real value; nothing about the token's structure reveals or derives from what it replaced, so there's no offline computation that gets you from token to source, only theft of the lookup table itself.

Masking and redaction don't try to preserve reversibility, they replace the sensitive portion with blanks or placeholders, `XXX-XX-1234`, the last four digits shown and the rest gone. It's the bluntest of the four and the hardest to get wrong: no algorithm to misapply, no salt to forget, just data genuinely no longer present in the output.

The pattern across all four: pick the technique based on what the attacker actually knows about your input space, not on which one sounds cryptographically serious. A one-way hash of a nine-digit number that anyone can enumerate offline isn't obfuscation. It's a lookup table you didn't build on purpose.

Written by Scott S. — Contact: scott@scottschlangen.com — scottslab.io — https://scottslab.io/posts/deidentification-anonymization-data-obfuscation