Revocable Privacy: Principles, Use Cases, and Technologies*

Wouter Lueks[1], Maarten H. Everts[2], and Jaap-Henk Hoepman[1]

[1] Radboud University, Nijmegen, The Netherlands
{lueks, jhh}@cs.ru.nl
[2] TNO, Netherlands Organisation for Applied Scientific Research
maarten.everts@tno.nl

Abstract. Security and privacy often seem to be at odds with one another. In this paper, we revisit the design principle of revocable privacy which guides the creation of systems that offer anonymity for people who do not violate a predefined rule, but can still have consequences for people who do violate the rule. We first improve the definition of revocable privacy by considering different types of sensors for users' actions and different types of consequences of violating the rules (for example blocking). Second, we explore some use cases that can benefit from a revocable privacy approach. For each of these, we derive the underlying abstract rule that users should follow. Finally, we describe existing techniques that can implement some of these abstract rules. These descriptions not only illustrate what can already be accomplished using revocable privacy, they also reveal directions for future research.

1 Introduction

Privacy and (homeland) security seem to be at odds with one another: it is a commonly held belief that we cannot strengthen one without weakening the other. And it seems security is winning. The governmental hunger for data—and its ability to actually gather these—seems bigger than ever. And who would argue against collection of these data? Surely we all want to stop terrorists, pedophiles and tax evaders. Yet, security versus privacy does not have to be a zero-sum game [15, 17]. Hoepman also argued that this contradiction between security and privacy is a false one, and that we can design systems that have privacy without neglecting security [11].

Hoepman introduced a design principle to create systems that have both security and privacy: *revocable privacy*. The core idea of revocable privacy arises from the realisation that it is not the data itself that is (or should be) important, but rather the

* This paper is based on our earlier technical report on revocable privacy [13]. The work described in this paper has been supported under the ICT theme of the Cooperation Programme of the 7th Framework Programme of the European Commission, GA number 318424 (FutureID) and the research program Sentinels (www.sentinels.nl) as project 'Revocable Privacy' (10532). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. This research is conducted within the Privacy and Identity Lab (PI.lab) and funded by SIDN.nl (http://www.sidn.nl).

violations of certain rules that manifest themselves in the data. Data related to people who do not violate any rules are irrelevant, and, in fact, these people should remain anonymous, as if no data on their behavior was ever collected. Revocable privacy is a design principle that ensures this property. Informally speaking, a system offers revocable privacy if users of the system are guaranteed to be anonymous except when they violate a predefined rule.

To ensure privacy, the system's anonymity guarantees cannot rely on policy and regulations alone. It is all too easy to ignore policy, to sidestep it, or to change it retroactively. As a result, data that was collected for one specific purpose can easily be reused for another—violating people's privacy. A key aspect of a system implementing revocable privacy is to prevent this type of function creep through technical means: it should not be possible to change the rules retroactively.

It is known that building such systems is possible. One example is the anonymous electronic cash system proposed by Chaum [6], which actually implements revocable privacy (although he did not use this term). Users have electronic coins, which they can spend as if they were physical coins, in effect making an untraceable digital payment system in which the users' privacy is guaranteed. However, to maintain security, this anonymity cannot be unconditional. If it were, it would allow misbehaving users to double-spend the digital coins without consequence. Instead, the revocable privacy aspect of the design guarantees that users are anonymous, as long as they spend the digital coins only once. When they do spend a coin twice, their identity can be recovered from the two transaction records of the two spendings. Any single transaction record, however, gives no information about the identity of the user.

In general, to ensure anonymity for rule-abiding users, data must be collected in a special manner. In Chaum's electronic cash system, the cut-and-choose paradigm is used to ensure that a single transaction record gives no information, whereas two reveal the identity of the culprit. Distributed encryption [12, 14] offers another method for creating threshold based rules. Data is collected for every event, but the user's identity is revealed only if she causes an event to happen at sufficiently many different locations.

While Chaum's electronic cash could be seen as such a scheme with a threshold of two, it differs significantly from distributed encryption. In the first, the user actively partakes in the transaction, whereas in the second, the user deliberately does not take part. As a result, these systems have different privacy guarantees and trust assumptions. These aspects of revocable privacy had not yet been explored.

In all previous work on revocable privacy [11–14], the focus was on identifying users who violate the rules. However, in some situations, such an approach might be too strong. For example, anonymity is the core property of Tor [8], so it should never be possible to deanonimize users. Yet, Tor can also be abused. In order to stop abuse, some approaches, like blacklistable anonymous credentials (BLAC), aim to block misbehaving users, rather than to identify them [18].[3]

Our first contribution, in Section 2, is to re-examine revocable privacy in a more general setting, where we consider the implications of different security models, and

[3] Nymble [19] is a related system that can be used to block misbehaving users. However, it relies on a trusted party that can make users linkable if they misbehave, so we do not consider it further in this paper.

explore ramifications of users' actions that are less invasive than simply identifying users, for example, blocking users and linking their actions.

Next, we explore and classify some use cases for revocable privacy in Section 3. We generalize the underlying rules of the use cases into abstract rules. These use cases illustrate that even if a user has violated a rule, she did not necessarily do something wrong. In fact, we will explore some systems where a violation only means that closer examination is necessary.

The abstract rules for the use cases make it possible to link them to specific techniques. Our final contribution, in Section 4, is to give a non-technical overview of existing techniques that can be used to implement revocable privacy. For each technique, we indicate which abstract rules it can implement. This not only shows which use cases we can already solve, but also highlights which abstract rules we cannot yet implement. We analyse the latter in Section 5 to reveal interesting new research directions. We also discuss some general limitations of revocable privacy. Finally, we conclude our paper in Section 6.

Revocable privacy is not a license for unchecked surveillance. The use cases explored in this report come from various sources. Some of them are real, others are purely hypothetical. In many cases the legality and/or morality of the situation described in the use case is debatable. We have included them for the sole purpose of investigating the types of rules a system with revocable privacy might need to implement in the future. Inclusion of a use case in this paper does not mean that we endorse it in any way.

2 Revisiting the concept of revocable privacy

In this section we will (re)define revocable privacy. We first explore what it means to be anonymous and what levels of anonymity exist.

2.1 Levels of anonymity

At first sight it may seem that anonymity is an all-or-nothing property: either you have it or you do not. This is false. There are many shades of anonymity, ranging from fully anonymous to fully identified. For example, users might be pseudonymous: their actions are known under a fixed identifier—the pseudonym—but it is not known which pseudonym belongs to which user. In fact, users may have different pseudonyms depending on the situation.

Pseudonymity is often even equated with anonymity. However, stronger forms of anonymity are possible. When a user's actions are unlinkable, it is impossible to determine whether two actions were performed by the same user or by different users. (This linking is trivial in a pseudonymous system.) When we say that a system is fully anonymous, we mean that it has this level of unlinkability.

We can traverse the range between fully anonymous to fully identified by adding pieces of information. Some have a small impact on anonymity, like gender, nationality or age. We can also add a pseudonym to make a user's action linkable within a specific

domain. The most natural pseudonym is one that does not change. Then all the user's actions will be linkable. However, we can also make pseudonyms that change frequently, and thus make the user's actions linkable within a short time period. Finally, some data, like social security numbers, license plates and bank account numbers, effectively make the user fully identified.

These ranges in anonymity have two consequences when dealing with revocable privacy. First, you can lose anonymity (because you violated a rule) without becoming fully identified. Second, it is better to see losing anonymity in relation to other participants in the system, as some systems may not offer full anonymity in the first place.

2.2 Improving the definition

Hoepman [11] originally defined revocable privacy as follows:

> "A system implements revocable privacy if the architecture of the system guarantees that personal data are revealed only if a predefined rule has been violated."

There are some problems with this specific definition. First, the rule explicitly mentions *personal* data. Companies, however, might have an equally big desire for protecting their corporate data (e.g., their business processes). Moreover, as we explored in the previous section, revealing personal data is not always necessary; there are other ways to lose anonymity.

The definition could also be extended to include cases where revealing a user's personal information could be positive to the user, rather than just negative, as we have discussed so far.[4] One example is privacy-friendly matching on a dating site, where you get each other's contact information only if the profiles match. However, we think that such systems should not be classified as having revocable privacy, as this makes the definition too broad, almost to the point of including all privacy enhancing technologies.

The second problem we have with this definition is that it is very easy to misread it and assume that if a person were to violate the rule then personal data are revealed. However, it does not say that. It states just that personal data *may* be revealed only if the rule is violated.

We incorporate these suggestions into the following revised definition of revocable privacy. We focus on anonymity rather than personal data and rephrase the rule to clarify that violating a rule does not necessarily imply the release of personal information.

Definition 1. *A system implements revocable privacy if the architecture of the system guarantees a predefined level of anonymity for a participant as long as she does not violate a predefined rule.*

As required, this definition does not say anything about the consequences when a participant does violate the rule. In practice there will be a consequence. If the system implements revocable privacy this usually means that the participant loses anonymity, but it can also mean that the participant is blocked from making further actions.

[4] In fact, we suggested this approach in our technical report [13].

2.3 Systems and rules

In the above definition, we consider the *system* as the environment with which the user interacts, and within which certain rules should be maintained. For example, in Chaum's electronic cash scheme, this system is the payment environment.

Rules are part of the system, and we require them to be predefined, including their parameters. For example, if the abstract rule is "A participant is allowed to cause an event at most t times", then the threshold t should be defined for every instantiation. This requirement prevents function creep and ensures clarity. If, instead, parameters should be configurable afterwards—for example, if certain criteria are not known in advance—the rule should explicitly state this.

We impose no other restrictions on the rules as this allows us to best capture the notion of 'anonymity until violation of the rules' that we see in many systems. In particular, we do not demand the rules to be known to the participants. While it is better that the rules are known, there might be circumstances where they must be kept secret.

We realize that the freedom in choosing rules (and keeping them secret) makes them very powerful. In fact, a rule might simply require all events to be output, or allow parameters to be set to non privacy-friendly values. Thus, careful scrutiny of the rules is of the utmost importance. The designers of the rules must ensure that the reduction in privacy that results from violating a rule is proportional to the detected behavior.

Ensuring proportionality is particularly important when violation of a rule does not necessarily imply that the participant is misbehaving. It may only be an indication of misbehavior (as in the canvas cutters use case, see Section 3.1) or even that the participant might be harmed (as in the detecting child abuse case, see Section 3.3).

2.4 Architecture of a System

What does it mean for the architecture of the system to protect the anonymity of well-behaving users? As we argued before, policies and procedures do not offer sufficient protection against function creep and misuse of the data in the future. We cannot assume that the raw data remain secure forever. Instead, we rely on the architecture of the system (the manner in which it is built, including the cryptography) to guarantee the anonymity of rule-abiding participants.

However, systems implementing revocable privacy cannot always offer unconditional anonymity either. It matters how the user's actions are observed within the system. For example, if the system sees what the user does, but chooses to forget it, we have to trust the system to actually do this. In this section, we explore the trust assumptions in a system implementing revocable privacy.

To determine these assumptions we examine how data is collected—is the identity of the user ever known?— and how it is stored. We consider three conceptually different methods. For reference, we first describe the traditional method where the user is known and the data is stored in the clear. In the second method, the user is still identified, but only processed information is stored. In the third, the user is never identified.

In all of these situations data resulting from the user's actions are stored. A final post-processing procedure, that is based on the rule, takes these data and outputs data such that a negative consequence for the participants can be effected. Usually, these

data will reduce the anonymity, but they might also be used just to block further access to the system—as is the case in BLAC. Both how the data is encoded and how it is post-processed depend on the rule. In a system with revocable privacy, it is not possible to change the rule and then reprocess old data (that was collected using a different rule) according to the new rule.

Plaintext logging For contrast, we first describe the obvious method for implementing a system with rules. Users are never anonymous with respect to the system. Every relevant action by the user—relevant with respect to the rules that are to be enforced—is stored together with the user's identity.

Violations of the rules are detected by checking the rules against the stored data. Since the user's identity is also stored, any consequence to the user's actions can immediately be enforced.

Any anonymity guarantees offered by such a traditional system rely on the policies and regulations that protect access to the stored data and govern the data retention policies. Hence, trust lies in the policies. Because this is not an architectural protection, we say that such a system does not offer revocable privacy.

One way to bolster the protection is to add one or more trusted third parties that decide if the rule is violated and then carry out the desired consequence. Hoepman [11] says such a system, which he calls of the *spread responsibility* type, does have revocable privacy. However, since the system does not enforce the rule we do not consider that to be the case in this paper. Instead, we focus solely on systems that, according to Hoepman, have a *self-enforcing architecture*, where the architecture determines if the anonymity guarantee can be weakened.

Non-interactive sensors with encoding The second method is a direct alternative to traditional methods. It drastically improves the anonymity guarantees, without requiring changes to the users of the system. As with plaintext logging, the actions of the participant and its identity are visible to the system, however, in contrast, they are never stored directly. Instead, a sensor (there can be many sensors in a system) observes these actions and identities, and then transforms them, based on the rule, into encrypted data. Only these encrypted data are stored.

The encryption method is special. There is no key that can be used to decrypt the data. Only when the encrypted data corresponds to a violation of the rule, can they be decrypted to produce some useful output.

To guarantee anonymity, we need to trust that the sensors behave as specified. In particular, we trust that sensors do not store their inputs. In addition, many sensors use private keys, in which case we trust them to keep these secret too. These private keys ensure that even if the sensors' outputs are deterministic (i.e., the sensors do not use randomness) an attacker cannot simply confirm a suspected event based on the stored data by simply calculating the same function as the sensor.

Despite these strict trust assumptions on the sensor, these systems can be very useful because they can be used as a drop-in replacement for traditional systems. In particular, they do not require any changes to the user's side. Of course, the sensors and the rest of

the system still need to be adapted to work with the encrypted data. In some sense, the sensors act on behalf of the user.

Mitigating the trust needed in the sensors. In some cases, like the threshold system that we describe in Section 4.1, the sensors are distributed. In this case, it may be possible that some are compromised, while the system as a whole still offers (some) anonymity.

Another approach that is useful in this setting is to make sure that the system is forward secure. Loosely speaking this would imply that if a sensor is compromised, it impacts only future events.

Interactive sensors In the third and final method, there is no sensor that simply observes the user; the user herself needs to be actively involved and interact with the sensor. The user usually keeps track of some secret information.

The advantage of this approach is that the user's identity is never known to the system, so the user's anonymity does not rely on the trustworthiness of parties within the system. The downside is that the user needs to interact with the sensors.

The sensor cannot rely on its own inputs to verify the correctness of the supplied information. Instead, this burden falls on the participant: she needs to convince the sensor that the supplied information is correct (even if the sensor does not know the content of the information).

3 Use Cases

We now present a number of use cases that could benefit from revocable privacy. These use cases are the result of interviews with security experts, internal discussions and privacy enhancing technologies literature. This overview is by no means exhaustive. Instead, it serves as a motivation for revocable privacy and as a source of insight into the abstract rules underlying these cases. We use these abstract rules to determine which cases we can already solve, and for which ones we need to develop new primitives.

We omit some of the use cases from our original analysis [13]. As we discussed in Section 2.2, we omit cases where the user would benefit from having its anonymity revealed. Other cases we omit because they are too vague or not interesting. Finally, we omit cases that simply give too much power to a government agency, even if only suspicious behavior would be detected.

We sort and categorise the use cases based on the type of rule that the system should enforce. The rules are roughly ordered by complexity. We start with three simple classes. The first class is that of threshold rules, where an event should not happen too often. The second class, containing predicate rules, consists of rules for logical combinations of simpler events. The third class covers cases where the rule encodes a human decision making element—for example, a judge signing a warrant.

Next, we consider two classes of more complicated rules. The first covers rules that are more complex than any of the aforementioned. For example, rules about flows on graphs (useful in tax situations) and about combining (private) information into the decision making process. The second covers rules that are fuzzy and would normally, even when no anonymity is required, involve machine learning and data mining techniques.

Table 1. An overview of the use cases and their sensor type, source, and applicable techniques. The sensor type is non-interactive (N-I), interactive (I) or both. The techniques are distributed encryption (DE, Section 4.1), n-times anonymous credentials (n-AA, Section 4.1), blacklistable anonymous credentials (BLAC, Section 4.2), group signatures (GS, Section 4.2) and secure multi-party computation (MPC, Section 5). A question mark indicates that we are not sure if this technique fully solves the problem. Biskup and Flegel proposed a system [2] to solve the cases marked with an asterisk (*). However, it requires the sensor to store a (partial) record of all events, it thus does not offer the anonymity that our definition of revocable privacy requires. We do not know of a solution for these cases that implements revocable privacy as we defined here.

Use case	Type	Source	Technique
Canvas cutters	N-I	Dutch National Police (KLPD)	DE
Object surveillance	N-I	Dutch National Police (KLPD)	(*)
Average speed checking	N-I	Dutch National Police (KLPD) & Lueks et al. [14]	DE
Anomalies in logs	N-I	Biskup and Flegel [2]	(*)
Sharing anon. resources	I	Camenisch et al. [4]	n-AA
No-show reservation	I	Internal discussion	Unknown
Electronic cash	I	Chaum [6]	n-AA
Social welfare fraud	both	Municipality of Groningen	Unknown
Terrorist activity	N-I	Internal discussion	Unknown
Child abuse	N-I	Internal discussion	DE
Anonymous editing	I	Tsang et al. [18]	BLAC
Deanonymizing comments	both	Interview	GS?
Wiretapping policy	N-I	Interview	Unknown
Riot control	N-I	Dutch National Police (KLPD)	Unknown
Money flow anomalies	I	Sharemind application [3]	MPC?
Object surveillance 2	N-I	Internal discussion	Unknown
Camera footage	N-I	Internal discussion and Sound Intelligence [16]	Unknown

For each of these classes we present several use cases. For every use case, we describe the case, extract an abstract rule and note the consequence of violating that rule. While the use cases focus on specific scenarios, the abstract rules generalize the rule within these scenarios. It ignores the scenario specific details. This makes it easier to determine which use cases have similar rules, and which techniques might be used to solve a use case using revocable privacy. Table 1 records the type of sensor, the source of the use case, and potential solutions.

3.1 Threshold rules

A threshold rule has the form "a certain action should be performed no more that k times within a certain time period". The most common consequence of violating the rule is to reveal the violator's identity, however, it is also possible to block the user. A threshold of one is possible in some of these scenarios. The following use cases work with threshold rules.

Canvas cutters This case, as well as the following two cases, focusses on detecting bad or suspicious behavior involving cars. As cars are generally not able to communicate with roadside equipment, we focus on the scenario where an automatic number plate recognition system reads the license plates of passing cars. This makes using a non-interactive sensor the only option.

Criminals frequently loot trucks parked at rest stops by cutting the canvas that protects the goods. One way to detect these criminals is to look for cars that enter several different rest stops within a couple of hours. These cars are suspicious. While false positives cannot be eliminated—e.g., police cars and road-side assistance vehicles may cause them as well—most hits will correspond to suspicious behavior.

Abstract rule Given n sensors at different locations, a participant should trigger at most threshold t different sensors within a given time period.
Consequence The system learns the identity of the participant.

Object surveillance Related to the previous problem is the problem of casing: criminals checking out a location, like a sensitive piece of infrastructure, multiple times. These criminals can be detected by looking for cars that pass by this location rather frequently. This case is not the same as the canvas cutters use case. In particular, here all events contribute to the threshold, whereas for the canvas cutters case the number of different locations of the events matters. Again, false positives cannot be eliminated.

Abstract rule Given one or more sensors at one locations, a participant should trigger at most threshold t sensors (counting repeats) within a given time period.
Consequence The system learns the identity of the participant.

Average speed checking Besides spot checking with a speed camera, some countries have deployed average speed checking systems which measure a car's speed along a stretch of road [14]. For spot checks it is immediately clear whether an observed car is speeding. However, average speed checking requires some form of storage to determine the time it took a car to traverse a stretch of road. Phrased as a revocable privacy problem: the system should output the license plates of cars that pass two measuring station—one in the beginning and one at the end—within a too short time period.

Abstract rule Given n sensors at different locations, a participant should trigger at most threshold t different sensors in any time period of a given length.
Consequence The system learns the identity of the participant.

Anomalies in logs Servers keep activity logs. These logs can be used to detect attacks. One example of such an attack are repeated log-in attempts from the same remote system. These are easy to spot in the logs as they all originate from the same system. However, it is usually not necessary keep the logs for all the authentic users.

By nature of the system (the remote systems are identified by IP address) we an use non-interactive sensors to detect which remote system makes frequent fraudulent login attempts. These systems can then be blocked.

Abstract rule Same as for object surveillance.
Consequence The system learns the identity of the participant.

Sharing anonymous resources Some systems give people anonymous access to a resource on the basis that they can prove something—e.g., that they have a license to a game, or that they are of a certain age. While this anonymity is good for the user, it also makes it trivial to share the access with any number of people without detection. To limit this sharing, people could be allowed only a limited number of accesses per time period. When this value is exceeded—it should be chosen in such a way that under normal use it is not—the identity of the presumed sharer is revealed or the presumed sharer is blocked from accessing the system.

As the user and the system already interact, we prefer interactive sensors.

Abstract rule A participant of the system can perform an action at most n times per time period.
Consequence The system learns the identity of the participant.

No-shows in anonymous reservations Consider anonymous reservations of resources like cinema seats, museum access or computing resources based on unlimited access subscriptions. Resources, however, are often scarce, making no-shows undesirable. If the system is fully anonymous, it is not possible to discourage no-shows. Instead, we would like to construct a system that either blocks or deanonymizes a user if she does not use a reservation, or fails to do so too often, but lets honest users be anonymous.

Abstract rule A participant of the system that reserves a resource may fail to claim this resource only t times.
Consequence The system learns the identity of the participant or the system block the participant from making further reservations.

Electronic cash As we discussed in the introduction, another common case for revocable privacy is electronic cash [6]. Users are given digital coins that they can spend anonymously. However, they are not allowed to spend the same coin twice. This form of electronic cash is a threshold system with a threshold of two.

As before, using a non-interactive sensor is not desirable as the user already interacts with the receiver of the coin when she is spending it.

Abstract rule A participant can perform an action (e.g., spend one coin) at most once.
Consequence The system learns the identity of the participant.

3.2 Predicate rules

Not all rules are as simple as limiting the occurrence of an event. In this section we consider a class of rules that combine different indicators, similar to logical formulas.

Social Welfare Fraud In the Netherlands people can receive social welfare when they are unemployed. The amount received depends on the number of people in the household. In particular, they receive less welfare when they share living expenses. Some people defraud the system by incorrectly reporting that they live alone.

To detect possible cases of fraud, the municipality of Groningen looked for people who received social welfare and who indicated living alone but had higher utility consumptions (water, gas, electricity and waste) than would correspond to a one person household.[5] This search required collecting information from different sources. Using revocable privacy, it would be possible to combine these data, and only recover the suspected violations.

Data can be supplied to the system either using non-interactive sensors (for example, the utility companies and the government) or directly by the cooperating welfare recipients (the system verifies that they behave honestly).

Abstract rule Let every participant have a set of associated data items, and let $\mathcal{P}$ be a predicate over these data items. The predicate must be false.
Consequence The system learns the identity of the participant.

Detecting terrorist activity Contrary to the canvas cutters use case, a lot of law-enforcement-like cases depend on combining various indicators to find criminals. One rather primitive example works as follows. A person who buys fertilizer, rents a van and scouts a government building in a short period of time may be planning to make and set off a bomb.

Any one of these events might be totally benign. It is only the combination that leads to suspicion. In practice, the rules may be more complicated and involve different data items. Usually, the actual actions and the identity of the person performing them are known, making non-interactive sensors the most obvious choice.

Abstract rule Same as for social welfare fraud.
Consequence The system learns the identity of the participant.

3.3 Decision rules

All previous rules depend only on the inputs they receive. Given these inputs, the outcome is clear. People do not take part in the decision making process. However, sometimes this decision process is essential. For example, we do not know how to codify the rule "posts should not be offensive." Such a rule is better suited for human decision making. In this section, we discuss a few rules that include human decision making.

Detecting child abuse This first rule is actually a threshold rule, but with human decisions as input. Professionals working with children, e.g., doctors and teachers, may suspect abuse. However, for fear of causing undue panic and because reports would become part of the child's record, they may decide not to report this. These concerns

[5] The original source, http://gemeente.groningen.nl/algemeen-nieuws/2010-1/sociale-dienst-spoort-bijna-driehonderd-gevallen-van-bijstandsfraude-op (Dutch, last accessed January 29, 2012), is currently unavailable. The same technique is mentioned on http://www.nu.nl/politiek/2670044/aanpak-bijstandsfraude-bestandskoppeling.html (Dutch, last accessed May 31, 2015)

would be alleviated if these reports could be made in such a way that a child's identity becomes available only when a predetermined number of professionals agree that a child might be abused. In this situation using an interactive sensor is truly undesirable as it would alert the child or its guardians to the suspicion of abuse.

Abstract rule Same as for canvas cutters.
Consequence The system learns the identity of the participant.

Blocking anonymous editing In the previous case it was essential that there was no interaction with the participant (the child). Here, we consider another case where interaction *is* possible: anonymously editing Wikipedia pages. Given the sensitive nature of some Wikipedia pages, it would be beneficial to allow anonymous edits. Yet, this anonymity can also facilitate abuse, and this abuse is usually not easily detected automatically. Yet, people are good at this task, in fact, Wikipedia is based on this principle.

To protect the system, an anonymous user should be blocked from making further edits if one or several of her edits have been classified as abusive. Even though a moderator can classify an edit as abusive, and thus block a user, she should never be able to recover the identity of the editor.

Abstract rule Participants are not allowed to perform more than t bad actions.
Consequence The system blocks the participant from performing further actions.

Deanonymizing comments Like edits on Wikipedia, some posts on an online bulletin board might be made anonymously. Another method of discouraging abusive comments is to actually reveal the identity of the author. However, since this decision is rather invasive, the identity of the author should only be revealed if a sufficient number of moderators agree to do so.

It is possible to build this system with a non-interactive sensor. However, the user already interacts with the system, so an interactive sensor provides better privacy.

Abstract rule Participants are not allowed to perform actions that are deemed bad by more than t moderators.
Consequence The system learns the identity of the participant.

Wiretapping policy Typically, law enforcement agencies require permission, for example from a judge or other authority, before they can legally tap phone and internet connections. However, this is enforced only by policy.

To increase privacy, telecom operators could send the requested data to law enforcement agencies in such a way that the agencies can only access this information after the required permission has been obtained. In this case, it is the telecom operator that acts as a non-interactive sensor.

Abstract rule Participants can perform actions. No trusted party decides that the participant's behavior is suspicious.
Consequence The system learns the future actions of the participant.

3.4 Complex rules

We now discuss rules that are more complex, for example because they operate on graphs and labeled data, or because they use auxiliary information that should be protected as well.

In principle, the rules in this class can be described by any deterministic computer program. However, to illustrate how hard some of these tasks can be, we also discuss fuzzy rules, based on for example machine learning, in the next section.

Riot control In 2009/2010 there were riots between two ethnic groups in Culemborg (a Dutch city). The police knew that the rioters might receive reinforcements from certain parts of the country. To prevent them from arriving, they wanted to detect these groups en route, and block the exits to Culemborg at the appropriate times.

To detect these groups, they automatically read license plates. If a group of more than four cars originating from the reinforcement area was detected on the highway, they closed the high-way exit.

Two things make this case interesting. First, the goal is not to deanonymise specific cars, but rather to detect a group of cars from a specific area. Second, in order to make this system work auxiliary information is required about where cars are registered.

Abstract rule A sensor should observe at most n objects (with associated data) satisfying a predicate $\mathcal{P}$ within a time period.
Consequence The system learns that a match has been found.

Money flow anomalies Some types of tax fraud manifest themselves in in discrepancies in money flows between companies. In particular, whatever company A claims to have sold to B should also be reported as bought from A by B. However, cash flows between companies also reveal strategies and other sensitive economical information that companies would rather not share. Instead of just sharing this information, the tax office and the companies could build a revocable privacy system where a company's name is revealed only if it incorrectly reported its cash flow.

Abstract rule Given a graph, with the participants represented by nodes and the edges representing money flows between them. Participants should report the flows over their adjacent edges correctly.
Consequence The system learns the identity of the participants.

3.5 Fuzzy rules

Until now we discussed situations where the participants are easy to recognize because they have a unique identifier (e.g., license plate, social security number, name). However, this is not the case, for example, when only video of a person is available. Even if it is possible to recognize people using facial recognition, we may still need to recognize suspicious behavior. This brings us to the realm of fuzzy and probabilistic computation. We consider this class separately because we suspect it is even harder to solve these cases using revocable privacy.

Object surveillance based on people This first use case is similar to the object surveillance case earlier, but with the twist we described in the previous paragraph: we have only video of the people in the system. We want to know if someone cases a location, but without the convenience of a fixed identifier.

In a system without revocable privacy, we could (maybe) collect facial features of all recorded people and determine how often they show up. To do this in a privacy friendly manner would require a system that can take faces as input, and keep track of how often a specific face has been seen. Furthermore, even if this works on features that are derived from the image, the original(s) are necessary to make future identification possible. This is why we classify this case as fuzzy.

Abstract rule As for the object surveillance rule, but now with video as input.
Consequence The system obtains a picture or video of the participant.

Retrieving camera footage after a crime Many cities install cameras to increase safety. One way to use these cameras is as a remote viewing tool, so that it is easier to monitor many locations at once. However, often the camera feeds are also stored in case something untoward happens later on. However, when nothing bad happens, the data can safely be discarded. The data is only stored to obtain more information after a crime has been detected.

If the system automatically detects bad situations (for example based on sounds [16]), the system could encode the data, and only release past records when a bad situation is detected. Hence, the system guarantees that the privacy sensitive recordings are kept and released only when necessary.

Relying on a human operator to make the decision to reveal footage, would put this use case are back in the decision-making class of cases.

Abstract rule Participants should behave properly on camera (as determined by the system or operator).
Consequence The system obtains footage around the violation.

4 Technologies

In this section, we review some technologies from the past twenty years that can be used to implement revocable privacy. Table 1 shows which techniques apply to which use cases.

4.1 Threshold primitives

We begin by discussing primitives that can be used to implement threshold rules.

Distributed Encryption The distributed encryption primitive [12, 14] was specifically designed to solve revocable privacy problems with a threshold rule, in particular, the canvas cutters scenario. As such, it describes how non-interactive sensors (ANPR stations at the rest stops) can encrypt messages (license plates) in such a way that only if

enough encryptions (by different stations) of the same message are available they can be combined to recover the original message.

Obviously, any corrupted sensor can encrypt any message of an attacker's choosing. So the system guarantees security only as long as not too many sensors are corrupted.

The distributed encryption primitive counts only events, while many of the use cases count events per time period. In most cases, it suffices to restart the system for every new time period. If it is required that no more than a number of events happen in any time interval of a given length, then it is necessary to start overlapping instances of the system. The extensions by Lueks et al. [14] make it efficient to do so and ensures forward secrecy: even if a sensor is corrupted, it cannot be used to obtain information about previous time periods.

Combining the encrypted messages to recover the messages is not very efficient; it is exponential in the threshold. However, if the number of messages is small and they can be enumerated, like for license plates, then another technique by Lueks et al. [14] allows the system to trade space for time, making it reasonably efficient.

n-times anonymous credentials Whereas the previous primitive uses non-interactive sensors, n-times anonymous credentials [4] allow participants to directly interact with the sensors. As a result, the trust assumptions are much weaker. The system gives every user a credential. The user can use this credential to anonymously authenticate n times per time period. If the user authenticates more often, its identity becomes known.

Effectively, the user can create n different (random) numbers per interval. If the user authenticates more often, she is bound to reuse one of the previous ones. This makes it easy and efficient to detect violations of the rule. Extensions make it also possible for a user to exceed the limit a couple of times (possibly in different time periods) before its identity is revealed. These schemes are a generalization of electronic cash schemes [1, 5, 6] where the limit is to spend every coin only once.

4.2 Decision primitives

We now discuss techniques that can be used to implement decision-based revocable privacy rules.

Blacklistable anonymous credentials Blacklistable anonymous credentials [18] make it possible for a service provider to block users from future authentication if the user misbehaved in an earlier session. To enable this, the user uses his (certified) private key to generate a new, random token for every authentication. These tokens are bound to the user (but, without the user's private key it is impossible to determine to which user they belong). In addition, the user proves that it did not generate any of the tokens that the service provider placed on the blacklist.

If the service provider later detects abuse, it can add that session's token to the blacklist. The corresponding user can then no longer prove that its tokens are not on the blacklist and loses access to the service. Alternatively, the user can prove that it has no more than n tokens on the blacklist, thus the system allows a few bad actions.

The complexity of this protocol is linear in the number of items on the revocation list, making it inefficient. Some techniques can be used to reduce the complexity [10].

Group Signatures with Distributed Management A group signature scheme allows members of a group to digitally sign documents on behalf of the group [7]. The signers are anonymous in the sense that it is known only that they belong to the group, not who they are. A special party, the tracing agent, can overcome this anonymity and determine who created a specific signature, thereby revealing the identity of the signer.

Already this scheme can be used to implement the simple rule that you lose your anonymity when the tracing agent decides that this should happen. However, in some sense we are then back to having a single, trusted third party. Instead, we can distribute the powers of the tracing agent. In a group signature scheme with a distributed tracing agent, several agents need to cooperate before the identity of the signer can be recovered [9]. As long as a decent subset of the tracing agents is trusted, the anonymity of the user's identities is guaranteed.

5 Analysis

In the preceding section, we reviewed some techniques that can be used for revocable privacy. Unfortunately, we do not know of existing primitives for many of the more complex rules. Only the threshold use cases are covered reasonably well by existing techniques. This suggests that there might be relatively simple techniques that work for the object surveillance and anomalies in logs use cases.

For decision-based rules, there are some existing techniques that help solve some of the use cases. This again suggests that we might be able to develop techniques for the remaining cases (deanonymizing comments and wiretapping policy).

Nevertheless, there are some very generic techniques that could help implement the remaining rules using the revocable privacy paradigm. First, the field of privacy-preserving data mining might help in solving some of the anomalies like social welfare fraud.

Finally, there is multi-party computation. This technique allows multiple parties, each with their own private input, to compute any shared function over the data. All inputs are kept private; only the output is shared. While this technique works, in theory, for any computation, including machine learning algorithms, it is also very computationally intensive. Yet, the Sharemind company successfully used their multi-party computation platform to solve several real-world problems on private data [3], one of which is the money flow problem.

5.1 Limitations

We briefly discuss two limitations of revocable privacy. The first is that to obtain better anonymity without losing security, we have to pay in computing power. This is especially the case for the non-interactive sensor techniques that we discussed. However, we think that this cost is often acceptable.

The other limitation stems from the fact that most use cases and all solutions describe positive effects. A participant performs an action, and as a result of doing so, can violate a rule. It seems much harder to handle negative events: what if you follow the rules if you do something, rather than not do it? For example, if you observe someone misbehaving, but fail to report it.

6 Conclusions

We have argued why revocable privacy is an important construct that can be used to increase the privacy of a system's participants while maintaining security. We have classified systems with revocable privacy into two classes: those with non-interactive sensors and those with interactive sensors. Furthermore, we have clarified the definition and have generalized it to include different types of consequences for violating the rules.

We have also explored use cases that benefit from a revocable privacy approach. This not only illustrates the usefulness of revocable privacy, but also allows us to compile some abstract rules that revocable privacy techniques should be able to implement. We have described some of these techniques and showed which problems they solve.

The comparison between the abstract rules and existing revocable privacy techniques identifies interesting directions of future work in the area of revocable privacy. Based on the fact that many threshold-based rules and decision-based rules already have corresponding primitives, we expect that the remaining ones may be solvable as well. Furthermore, we identify whole classes of more challenging research direction in finding techniques for the other use cases that lack corresponding techniques, most notably social welfare fraud detection, riot control, and object surveillance based on people.

References

1. Au, M.H., Chow, S.S.M., Susilo, W.: Short E-Cash. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3797, pp. 332–346. Springer (2005)
2. Biskup, J., Flegel, U.: Transaction-Based Pseudonyms in Audit Data for Privacy Respecting Intrusion Detection. In: Debar, H., Mé, L., Wu, S.F. (eds.) Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000, Toulouse, France, October 2-4, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1907, pp. 28–48. Springer (2000)
3. Bogdanov, D., Jõemets, M., Siim, S., Vaht, M.: How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation. In: Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers. LNCS, Springer (2015), to appear
4. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006. pp. 201–210. ACM (2006)
5. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 302–321. Springer (2005)
6. Chaum, D.: Blind Signatures for Untraceable Payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982. pp. 199–203. Plenum Press, New York (1982)

7. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. Lecture Notes in Computer Science, vol. 547, pp. 257–265. Springer (1991)
8. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The Second-Generation Onion Router. In: Blaze, M. (ed.) Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA. pp. 303–320. USENIX (2004)
9. Ghadafi, E.: Efficient Distributed Tag-Based Encryption and Its Application to Group Signatures with Efficient Distributed Traceability. In: Aranha, D.F., Menezes, A. (eds.) Progress in Cryptology - LATINCRYPT 2014 - Third International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, September 17-19, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8895, pp. 327–347. Springer (2014)
10. Henry, R., Goldberg, I.: Thinking inside the BLAC box: smarter protocols for faster anonymous blacklisting. In: Sadeghi, A., Foresti, S. (eds.) Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013. pp. 71–82. ACM (2013), http://doi.acm.org/10.1145/2517840.2517855
11. Hoepman, J.H.: Revocable Privacy. ENISA Quarterly Review 5(2) (Jun 2009)
12. Hoepman, J., Galindo, D.: Non-interactive Distributed Encryption: A New Primitive for Revocable Privacy. In: Chen, Y., Vaidya, J. (eds.) Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011. pp. 81–92. ACM (2011)
13. Lueks, W., Everts, M.H., Hoepman, J.H.: Revocable Privacy 2012 – use cases. Tech. Rep. 35627, TNO (2012)
14. Lueks, W., Hoepman, J., Kursawe, K.: Forward-Secure Distributed Encryption. In: Cristofaro, E.D., Murdoch, S.J. (eds.) Privacy Enhancing Technologies - 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8555, pp. 123–142. Springer (2014)
15. Schneier, B.: What Our Top Spy Doesn't Get: Security and Privacy Aren't Opposites. Wired (Jan 2008)
16. Sound Intelligence: Sigard, aggression detection, http://www.soundintel.com/uploads/pdf/UK/Sound%20Intelligence%20Brochure%20%28EN%29.pdf Last accessed: May 31, 2015
17. Stadler, M.: Cryptographic Protocols for Revocable Privacy. Ph.D. thesis, Swiss Federal Institute of Technology, Zürich (1996)
18. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. pp. 72–81. ACM (2007)
19. Tsang, P.P., Kapadia, A., Cornelius, C., Smith, S.W.: Nymble: Blocking Misbehaving Users in Anonymizing Networks. IEEE Trans. Dependable Sec. Comput. 8(2), 256–269 (2011)