Control, Confidentiality, and the Right to be Forgotten*

Aloni Cohen
aloni@uchicago.edu
University of Chicago

Adam Smith
ads22@bu.edu
Boston University

Marika Swanberg
marikas@bu.edu
Boston University

Prashant Nalini Vasudevan
prashant@comp.nus.edu.sg
National University of Singapore

ABSTRACT

Recent digital rights frameworks give users the right to *delete* their data from systems that store and process their personal information (e.g., the "right to be forgotten" in the GDPR).

How should deletion be formalized in complex systems that interact with many users and store derivative information? We argue that prior approaches fall short. Definitions of *machine unlearning* [6] are too narrowly scoped and do not apply to general interactive settings. The natural approach of *deletion-as-confidentiality* [15] is too restrictive: by requiring secrecy of deleted data, it rules out social functionalities.

We propose a new formalism: *deletion-as-control*. It allows users' data to be freely used before deletion, while also imposing a meaningful requirement after deletion—thereby giving users more control.

Deletion-as-control provides new ways of achieving deletion in diverse settings. We apply it to social functionalities, and give a new unified view of various machine unlearning definitions from the literature. This is done by way of a new adaptive generalization of history independence.

Deletion-as-control also provides a new approach to the goal of *machine unlearning*, that is, to maintaining a model while honoring users' deletion requests. We show that publishing a sequence of updated models that are differentially private under continual release satisfies deletion-as-control. The accuracy of such an algorithm does not depend on the number of deleted points, in contrast to the machine unlearning literature.

CCS CONCEPTS

• **Security and privacy** → *Cryptography*; **Privacy protections**; • **Social and professional topics** → **Privacy policies**.

KEYWORDS

deletion, machine unlearning, history independence, differential privacy

*A full version of this extended abstract appears on arXiv [10].

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11…$15.00
https://doi.org/10.1145/3576915.3616585

ACM Reference Format:
Aloni Cohen, Adam Smith, Marika Swanberg, and Prashant Nalini Vasudevan. 2023. Control, Confidentiality, and the Right to be Forgotten. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3616585

1 INTRODUCTION

The long-term storage of modern data collection carries serious risks, including often-surprising disclosures [7, 9, 14, 26, 33], manipulation, and epistemic bubbles. The permanence of our digital footprints can also chill expression, with every word weighed against the risk of out-of-context blowback in the future.

Data protection laws around the world have begun to challenge this permanence. The EU's General Data Protection Regulation provides an individual *data subject* the right to request "the erasure of personal data concerning him or her" and delineates when a *data controller* must oblige. California followed suit in 2020, and similar rights take effect in Virginia, Colorado, Connecticut, and Utah in 2023 [31].

In the modern data ecosystem, however, it is not easy to articulate what constitutes the "erasure" of personal data. Data is not merely stored in databases—it is used to train machine learning models, compute and publish statistics, and drive decisions. Such complexity and nuance challenges simplistic thinking about erasure, and the sheer number of ways data are used precludes case-by-case reasoning about erasure compliance.

Giving users more control over data is today a central policy goal. Decades of cryptography has given us good definitional tools for reasoning about non-disclosure of data—enabling the development of technical solutions, informing policy decisions, and influencing practice. But we lack similar tools for reasoning about control over data and, in particular, deletion. While there has been a flurry of recent work on so-called *machine unlearning* [5, 6, 16, 23, 36, 39, …], often directly motivated by legal compliance, there remain basic gaps in our understanding.

This paper sheds light on what data deletion means in complex data processing scenarios, and how to achieve it. We provide a formulation that unifies and generalizes previous technical approaches that only partially answer this question. Though we make no attempt to strictly adhere to any specific legal right to erasure, we aim to incorporate more of its contours than prior technical work on erasure.

Our new formulation, called *deletion-as-control*, requires that after an individual Alice requests erasure, the data controller's future behavior and internal state should not depend on Alice's

data *except insofar as that data has already affected other parties in the world*. In this way, Alice's autonomy need not require secrecy; she has a say about how her data is used regardless of past or future disclosure.

Our definition meaningfully captures a variety of data controllers, including ones facilitating social interactions and maintaining accurate predictive models. In contrast, prior approaches yield a patchwork of, at times, contradictory and counterintuitive interpretations of erasure.

1.1 Touchstone Examples

In order to understand our and prior approaches to data deletion, it is helpful to have in mind a few concrete examples of functionalities that separate our approaches from prior ones. We describe four touchstone functionalities, then briefly discuss how they relate to prior approaches and our new notion.

Private Cloud Storage. Users can upload files for cloud storage and future download. Only the originating user may download a file and the files are never used in any other way. The existence of files is only ever made known to the originating user, and the controller publishes no other information.

Public Bulletin Board. Users can submit posts to the public bulletin board. The bulletin board simply displays all user posts currently in the controller's internal storage, with no other functionalities (e.g. responding, messaging, etc.).

Batch Machine Learning. Users contribute data during some collection period. At the end of the period, the data controller trains a predictive model on the resulting dataset. The data controller then publishes the model.

Public Directory + Usage Statistics. Users upload their name and phone number to be listed in a directory. The data controller allows anybody to search for a listing in the directory, and each week reports a count of the number of distinct users that have looked up a phone number so far. (Other statistics are possible too—the weekly count of new users, say.)

The touchstone examples and prior approaches. Figure 1 summarizes how the touchstone functionalities fare under deletion-as-control and under three prior approaches to defining deletion: *deletion-as-confidentiality*, *machine unlearning*, and *simulatable deletion* (Section 1.4).[1] Together, the touchstone functionalities illustrate that prior approaches constitute an inconsistent patchwork, each falling short on at least two of the examples.

Deletion-as-confidentiality [15] is over-restrictive. Briefly, it requires that third parties cannot distinguish whether a data subject Alice requested erasure from the controller or simply never interacted with the controller in the first place. This implies, among other things, that Alice's data is kept confidential from all other parties even if Alice never requests its erasure. This confidentiality-style approach is well-suited for Private Cloud Storage, but deletion-as-confidentiality precludes inherently social functionalities, like the Bulletin Board and Directory. No controller implementing these functionalities could ever satisfy deletion-as-confidentiality. Why would Alice post messages if they could never be made public?

On the other hand, machine unlearning—even in its strongest incarnation, due to Gupta et al. [23]—is too narrowly scoped. It is specialized to the setting of machine learning and does not consider general interactive functionalities. The definition is not applicable to the Cloud Storage, Bulletin Board, and Directory functionalities. Definitions from the machine unlearning literature are meaningful for the Batch Machine Learning functionality, where they correspond to versions of *history independence*.

Even where multiple definitions are meaningful, they may impose different requirements. For example, both deletion-as-confidentiality and deletion-as-control admit implementations of Batch Machine Learning that are *persistent* in that the published models never need to be updated. On the other hand, history independence requires that any useful model be updated after enough deletion requests.

The touchstone examples and deletion-as-control. We show that each of the touchstones can be implemented in a manner that satisfies our new notion, deletion-as-control.

Private Cloud Storage. To remove a user, the controller deletes all the user's files from its internal storage. Such a controller satisfies deletion-as-control if its data structures are *history independent* (Corollary 3.8).

Public Bulletin Board. To remove a user, the controller deletes all of the user's posts from its internal storage and, as a result, from the public-facing bulletin board. As with cloud storage, such a controller satisfies deletion-as-control if its data structures are history independent (Corollary 3.8).

Batch Machine Learning. Deletion-as-control is achieved if the dataset is deleted after training and training is done with differential privacy, e.g., using DP-SGD [3] (Corollary 5.2). To remove a user, the controller does nothing—it simply ignores deletion requests. The resulting deletion guarantee is parameterized by the privacy parameters ϵ and δ.

Public Directory + Usage Statistics. Deletion-as-control can be achieved by combining differential privacy and history independence (Corollary 6.5). The statistics are computed using a mechanism that satisfies a stringent form of DP—pan-privacy under continual release—while the public directly is implemented using a history independent data structure. To remove a user, the controller deletes their listing from the public directory and its associated data structures, but leaves the data structures for the DP statistics unaltered.

1.2 Contributions

Defining deletion-as-control. Our primary contribution is a formalization of *deletion-as-control*, an important step towards providing individuals greater control over the use of personal data. The new notion applies to general data controllers and interaction patterns among parties, building on the modeling of [15]. As described below, it unifies existing approaches within a coherent framework and captures all the touchstone examples of Section 1.1.

The goal embodied by deletion-as-control is not so much to hide data from others as to exercise control over how the data is used. Until Alice requests erasure, deletion-as-control should not limit the controller's usage of her data. But after erasure, the data

[1] We introduce the terms *deletion-as-confidentiality* and *simulatable deletion* for the definitions of [15] and [17], respectively, to more clearly distinguish them from each other and from deletion-as-control. [15] use the term *deletion-compliance*; [17] use *strong deletion-compliance* and *weak deletion-compliance*, respectively.

	Private Cloud Storage	Public Bulletin Board	Batch Machine Learning	Public Directory + Usage Stats
Deletion-as-Confidentiality [15]	✓	⊤	✓	⊤
Machine unlearning [23, among others]	✗	✗	✓	✗
Simulatable deletion [17]	✓	⊥	⊥	⊥
Deletion-as-control (this work)	✓ (Cor. 3.8)	✓ (Cor. 3.8)	✓ (Cor. 5.2)	✓ (Cor. 6.5)

✓: Definition is satisfied by implementations with meaningful deletion guarantees.
⊥: Definition is under-restrictive: allows *vacuous* implementations with no meaningful deletion of any kind.
⊤: Definition is over-restrictive: no implementation of the functionality satisfies the definition.
✗: Definition does not apply to the functionality.

Figure 1: Application of deletion definitions to touchstone functionalities.

controller's future behavior and internal state should not depend on Alice's data *except insofar as that data has already affected other parties in the world*. In this way, Alice's autonomy need not imply secrecy; she has a say about how her data is used regardless of past or future disclosure.

Our approach provides new ways of achieving meaningful deletion in diverse settings.

Capturing social functionalities via history independence. Deletion-as-control applies to a wide range of controllers that provide "social" functionalities where prior approaches fall flat (e.g., the Public Bulletin Board touchstone). Along the way, we give a new unified view of the various machine unlearning definitions from the literature.

Both flow from a theorem roughly stating that deletion-as-control is implied by *adaptive history independence*, a generalization of the cryptographic notion of *history independence* [29, 30] that we introduce. An implementation of a data structure is history independent if its memory representation reveals nothing more than the logical state of the data structure. That history independence is related to deletion is intuitive, and appears in [15, 17]. Machine unlearning imposes a similar requirement in the specific context of machine learning. Oversimplifying, a learned model (akin to the memory representation) must reveal nothing more than a model retrained from scratch (akin to the logical state). We make these connections precise.

New algorithms for machine learning via differential privacy. Deletion-as-control provides a new approach for machine learning in the face of modern data rights. Very roughly, differential privacy (DP) provides deletion-as-control for free. Intuitively, if a person has (approximately) no impact on a trained model, mitigating that impact is trivial. In particular, if using an adaptive pan-private algorithm to maintain the model, it does not need to be updated in response to deletion requests, unlike machine unlearning algorithms. For the first time, this approach enables a meaningful deletion guarantee while bounding the worst-case loss compared to deletion-free learning.

Specifically, we describe two ways of compiling DP mechanisms into controllers satisfying deletion-as-control. The first applies to DP mechanisms that are run in a batch setting on a single, centralized dataset (e.g., the Batch Machine Learning touchstone). The second applies to mechanisms satisfying an adaptive variant of *pan-privacy under continual release* [8, 13, 25], including controllers that periodically update a model on an ongoing basis. The compilation

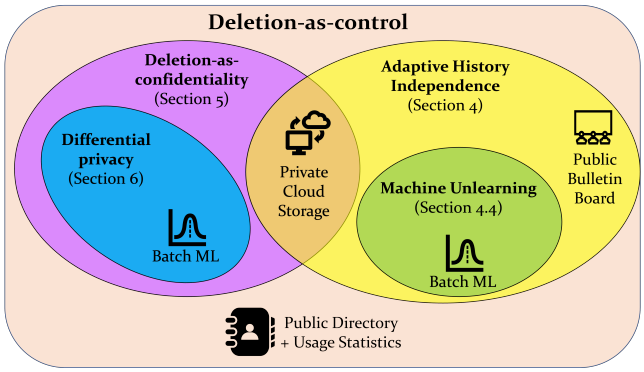

Figure 2: Overview of the relationships between definitions and the roles of specific example controllers. Each oval represents a definition, and each icon represents an implementation of a touchstone functionality satisfying the corresponding definition. Oval containment corresponds to an implication between the definitions, though only roughly and subject to technicalities that we do not attempt to capture in this figure.

from differential privacy is by way of deletion-as-confidentiality Garg et al. [15], which we prove implies deletion-as-control.

We combine this result with existing algorithms for private learning under continual release [27] to obtain new controllers that maintain a model with accuracy essentially identical to that of a model trained on the entire set of added records.

Capturing complex mechanisms via composition. We show that deletion-as-control captures more functionalities than those collectively captured by history independence, differential privacy, and deletion-as-confidentiality. Specifically, we show how to implement the Public Directory + Usage Statistics touchstone, a functionality that cannot satisfy any of the above three properties. To do so, we prove that deletion-as-control enjoys a limited form of *parallel composition*.

1.3 Defining deletion-as-control

We define deletion-as-control in a way that allows arbitrary use of a person's data before deletion, but not after. The challenge is to provide meaningful privacy guarantees despite this feature, even in a general setting with an adaptive and randomized data controller,

data subject (Alice), and an environment (representing all parties other than the data controller and Alice).

Our proposal is that after deletion, the data controller should be able to produce a plausible alternate explanation for its current state without appealing to Alice's participation. More specifically, *the state of the controller after deletion can be plausibly attributed to the interaction between the controller and the environment alone.* By this we mean that the state is about as likely—with probability taken over the controller's random coins—in the real world as in the hypothetical 'ideal' world where the environment's messages to the controller are unchanged but where Alice does not interact with the controller whatsoever. The result is that, after Alice's deletion, the controller's subsequent states depend on the interaction with Alice only insofar as that interaction affected other parties' interactions with the controller.

Importantly, we only require that the controller's state is plausible in the ideal world *given* the messages sent by the environment. *We do not require the environment's messages themselves to be plausible in the ideal world.* For example, suppose that on a public bulletin board, Bob simply copies and reposts Alice's posts. Bob's messages in the ideal world would still contain the content of Alice's posts, even though Alice is completely absent in the hypothetical ideal world. This is unavoidable—we want to allow the controller and the environment to use the subject's data arbitrarily before deletion. So the environment's queries may depend on the data subject's inputs, directly or indirectly.

In a bit more detail, our definition compares real and ideal worlds defined in a non-standard way.[2] The real world execution, denoted $\langle \mathsf{C}, \mathsf{E}, \mathsf{Y} \rangle$, involves three parties: a data controller C, a special data subject Y, and an environment E representing all other parties. (The notation $\langle \cdots \rangle$ denotes the transcript of an execution between the parties listed.) The execution ends after Y requests deletion and C processes the request. The real execution specifies (i) C's state $\mathsf{state}_{\mathsf{C}}$, (ii) the *queries* $\vec{\mathsf{q}}_{\mathsf{E}}$ sent by E to C, and (iii) the randomness R_{C} used by C. The ideal world execution, denoted $\langle \mathsf{C}(R'_{\mathsf{C}}), \mathsf{D}(\vec{\mathsf{q}}_{\mathsf{E}}) \rangle$, involves the same controller C and a dummy environment D that simply replays the queries $\vec{\mathsf{q}}_{\mathsf{E}}$. The controller's ideal world randomness R'_{C} is sampled by a simulator $\mathsf{Sim}(\vec{\mathsf{q}}_{\mathsf{E}}, R_{\mathsf{C}}, \mathsf{state}_{\mathsf{C}})$. The ideal execution specifies (i) C's ideal state $\mathsf{state}'_{\mathsf{C}}$, and (ii) the simulated randomness R'_{C} used by C.

Definition 1.1 ((ϵ, δ)-deletion-as-Control (simplified))**.** *Given* C, E, Y, *and* Sim, *consider the following experiment. Sample* $R_{\mathsf{C}} \leftarrow \{0,1\}^{\mathbb{N}}$*; run the real execution* $(\vec{\mathsf{q}}_{\mathsf{E}}, \mathsf{state}_{\mathsf{C}}) \leftarrow \langle \mathsf{C}(R_{\mathsf{C}}), E, \mathsf{Y} \rangle$*; sample* $R'_{\mathsf{C}} \leftarrow \mathsf{Sim}(\vec{\mathsf{q}}_{\mathsf{E}}, R_{\mathsf{C}}, \mathsf{state}_{\mathsf{C}})$*; and run the ideal execution* $(\vec{\mathsf{q}}'_{\mathsf{E}}, \mathsf{state}'_{\mathsf{C}}) \leftarrow \langle \mathsf{C}(R'_{\mathsf{C}}), \mathsf{D}(\vec{\mathsf{q}}_{\mathsf{E}}) \rangle$.

We say a controller C *is* (ϵ, δ)-deletion-as-control compliant *if there exists* Sim *such that for all* E *and* Y*: (i)* $R'_{\mathsf{C}} \overset{\epsilon,\delta}{\approx} R_{\mathsf{C}}$*; (ii)* $\mathsf{state}'_{\mathsf{C}} = \mathsf{state}_{\mathsf{C}}$ *with probability at least* $1 - \delta$.

The notation $\overset{\epsilon,\delta}{\approx}$ denotes approximate indistinguishability parameterized by ϵ and δ (as in the definition of differential privacy). We give a complete version of Definition 1.1 in Section 2.

[2]The non-real world is more 'hypothetical' or 'counter-factual' than 'ideal.' Regardless, we use the term 'ideal world' for continuity with [15] and decades of cryptography.

1.4 Prior Work

We give a brief discussion of prior definitions of deletion. Machine unlearning and deletion-as-confidentiality are discussed in detail in Sections 3.4 and 4, respectively.

Deletion-as-confidentiality. Garg et al. [15] define *deletion-as-confidentiality.*[3] It requires that the deleted data subject Alice leaves (approximately) no trace: the whole view of the environment along with the state of the controller after deletion should be as if Alice never existed. As a result, no third party may ever learn of Alice's presence — even if she never requests deletion. The strength of this definition is its strong, intuitive, interpretable guarantee.

But deletion-as-confidentiality is too restrictive (Figure 1). The stringent indistinguishability requirement precludes any functionality where users learn about each other. Implementations of the Private Cloud Storage and Batch Machine Learning functionalities can satisfy deletion-as-confidentiality, using history independence (cf. Section 3). But the Bulletin Board and Directory are ruled out. If Bob ever looks up Alice's messages, the confidentiality required by the definition is impossible.

Simulatable deletion. Godin and Lamontagne [17] introduce *simulatable deletion* as a relaxation of deletion-as-confidentiality, motivated by the observation that deletion-as-confidentiality rules out social functionalities.[4] Roughly, simulatable deletion requires that after a data subject Alice is deleted, the resulting state of the controller is simulatable given the environment's view. This means that any information about Alice that is present in the controller's state is already present in the view of other parties.

Simulatable deletion is too permissive. A controller may indefinitely retain any information that has ever been shared with any third party. For example, the Public Bulletin Board need not delete Alice's posts if they have ever been read![5] As a result, simulatable deletion is essentially vacuous for functionalities where the controller's state need not be kept secret. The controller can simply publish its state, making simulation trivial. Turning to our touchstone examples (Section 1.1), while simulatable deletion imposes a meaningful requirement for the Private Cloud Storage functionality, it allows implementations with no meaningful deletion of any kind for the other three functionalities.

Machine unlearning. This recent line of work specializes the question of deletion to the setting of machine learning [5, 6, 16, 23, 36, 39, ...]. Given a model $\vec{\theta} \leftarrow \mathsf{Learn}(\vec{x})$ and a data point $x^* \in \vec{x}$, that literature requires sampling a new model $\theta' \leftarrow \mathsf{Unlearn}(\vec{\theta}, x^*, \vec{x})$ approximately from the distribution $\mathsf{Learn}(\vec{x} \setminus \{x^*\})$ (i.e., approximating retraining from scratch). As we explain in Section 3.4, these definitions correspond to versions of *history independence.*

[3]Deletion-as-confidentiality is called *deletion-compliance* in [15] and *strong deletion-compliance* in [17].

[4]Simulatable deletion is called *weak deletion-compliance* in [17].

[5]Godin and Lamontagne [17] actually consider a version of the Bulletin Board functionality for which simulatable deletion is meaningful. Crucially, their functionality does not track whether a post has been read. Hence their controller must actually delete Alice's posts. But if the bulletin board keeps read receipts or actively pushes new messages out to users, say, it would not have to delete these posts.

A drawback of machine unlearning is that it specialized to the setting of machine learning. It does not apply to general data controllers and interaction patterns among parties, including the Cloud Storage, Bulletin Board, and Directory functionalities (Figure 1).

History independence-style definitions of machine unlearning do impose a meaningful requirement for the Batch Machine Learning functionality. In fact history independence is a conceptually stricter requirement than deletion-as-control, setting aside many technicalities (Section 3.4). Differentially private (DP) learning illustrates the difference. Roughly, DP learning provides deletion-as-control for free; the resulting model can be published once and never updated. In contrast, history independence *require* updating the model when there are many deletions. To see why, consider $\vec{x}$ of size n and suppose all n people request deletion. These definitions would require the final model θ^* be essentially trivial—it should perform about as well as a the model θ_0 trained on an empty dataset. With the DP learner described above, θ^* performs just as well as the initial model θ.

1.5 Paper Structure

In Section 2 we define deletion-as-control. In Section 3 we define Adaptive History Independence and prove that it implies deletion-as-control (Theorem 3.6). We also explain that definitions in the machine unlearning literature can be seen as a special case of Adaptive History Independence. In Section 4 we show that deletion-as-confidentiality is a strengthening of our definition (Theorem 4.3). In Section 5, we relate differential privacy to our definition (Theorem 5.3) and in the process we define adaptive pan-private in the continual release model. Lastly, in Section 6 we prove a narrow composition result for our definition. Additional details and complete proofs can be found in the full version of this paper [10].

2 DELETION-AS-CONTROL

We define deletion-as-control in a way that allows arbitrary use of a person's data before deletion, but not after. Under such a definition, an adversary might completely learn the data before it is deleted, and even make it available after it is deleted! The challenge is to provide a meaningful guarantee despite this limitation, even in a general setting with adaptive and randomized data controllers, data subjects, and environments (representing all parties other than the data controller and distinguished data subject).

2.1 (ϵ, δ)-indistinguishability

We consider a notion of similarity of distributions closely related to differential privacy .

Definition 2.1. *Given parameter $\epsilon \geq 0$ and $\delta \in [0, 1)$, we say two probability distributions P and Q on the same set $\mathcal{X}$ (with the same σ-algebra of events $\Sigma_{\mathcal{X}}$) are* (ϵ, δ)-indistinguishable *and write $P \overset{\epsilon,\delta}{\approx} Q$ if, for every event $E \in \Sigma_{\mathcal{X}}$,*

$$P(E) \leq e^{\epsilon} Q(E) + \delta \text{ and } Q(E) \leq e^{\epsilon} P(E) + \delta\,.$$

Slightly overloading this notation, we say two random variables X and Y taking values in the same measurable space $(\mathcal{X}, \Sigma_{\mathcal{X}})$ are (ϵ, δ)-indistinguishable, *denoted $X \overset{\epsilon,\delta}{\approx} Y$ if, for every event $E \in \Sigma_{\mathcal{X}}$,*

$$\Pr[X \in E] \leq e^{\epsilon} \Pr[Y \in E] + \delta \text{ and,}$$
$$\Pr[Y \in E] \leq e^{\epsilon} \Pr[X \in E] + \delta\,.$$

Because algorithms in our model run in unbounded time, their (countably infinite) random tapes belong to an uncountably infinite set. This means that not all sets of random tapes have well-defined probability. The σ-algebra $\Sigma_{\mathcal{X}}$ captures the set of events E for which $P(E)$ and $Q(E)$ are defined. This issue does not affect most proofs and definitions; we only make the σ-algebra of events explicit when necessary.

In our case, the set $\mathcal{X}$ of random tapes for a single machine is $\{0,1\}^{\mathbb{N}}$. Any execution that terminates reads only a finite prefix of the tape, and so the natural σ-algebra $\Sigma_{\mathcal{X}}$ is the smallest one containing the sets $E_w = \left\{w \| x : x \in \{0,1\}^{\mathbb{N}}\right\}$ for all $w \in \{0,1\}^*$, where $\|$ denotes string concatenation (that is, E_w is the set of infinite tapes with a particular finite prefix w). $\Sigma_{\mathcal{X}}$ contains every event E that depends on only finitely many bits of the tape. This is the standard σ-algebra for an infinite set of fair coin flips (see, e.g., [35]).

2.2 Parties and simplified execution model

Our definition uses the real/ideal cryptography, though not with a typical indistinguishability criterion.

Complete details are in the full version [10].

The real world execution, denoted $\langle \mathsf{C}, \mathsf{E}, \mathsf{Y} \rangle$, involves three (possibly randomized) parties: a data controller C, an environment E, and a special data subject Y. The real interaction is arbitrary, limited only by the execution model described below. Parties have authenticated channels over which they may interact freely. While Y has only a single channel to C, the environment has an unbounded number of channels (representing unbounded additional parties). While these channels are authenticated, the controller cannot distinguish the single channel to Y from those to E. The interaction continues until the data subject Y requests deletion, ending after the data controller C processes the request. We denote by $\text{state}_{\mathsf{C}}$ the final state of the controller.

The ideal world execution, denoted $\langle \mathsf{C}, \mathsf{D} \rangle$, involves the interaction of the same controller C as well as a dummy environment D. D takes as input the transcript from the real execution, denoted τ, and simply replays only E's queries, denoted $\vec{\mathsf{q}}_{\mathsf{E}}$. If $\vec{\mathsf{q}}_{\mathsf{E}}$ is empty, then D terminates without sending messages. Observe that C's responses and state in the ideal world are not fixed. They depend on C's ideal-world randomness, denoted R'_{C}. Moreover, the ideal interaction is defined relative to a particular instantiation of the real world interaction. In particular, the queries $\vec{\mathsf{q}}_{\mathsf{E}}$ may depend on C's real-world randomness, denoted R_{C}.

The controller's real-world randomness R_{C} consists of infinitely-many random bits sampled uniformly at random from $\{0,1\}^{\mathbb{N}}$. We denote this distribution $\mathcal{U}$. In the ideal world, a simulator Sim takes as input $(\vec{\mathsf{q}}_{\mathsf{E}}, R_{\mathsf{C}}, \text{state}_{\mathsf{C}})$ and generates C's ideal-world randomness R'_{C}. When we wish to emphasize the controller's randomness in the execution, we write $\langle \mathsf{C}(R_{\mathsf{C}}), \mathsf{E}, \mathsf{Y} \rangle$ and $\langle \mathsf{C}(R'_{\mathsf{C}}), \mathsf{D} \rangle$.

An execution involves parties sending messages to each other until some termination condition is reached. Starting with E (real) or D (ideal), parties get activated when they receive a message, and deactivated when they send a message. Only a single party is active at a time. Parties communicate over authenticated channels. Because E represents all users besides the distinguished data subject Y, E has many distinct channels to C. Importantly, authentication allows parties to know on which channel a message was received, but not which party (i.e., E or Y) is on the other end of that channel. Each party is initialized with a uniform random tape which may only be read *once* over the course of the whole execution. If a party wishes to re-use bits from its randomness tape, it must store them in its internal state.

The real execution $\langle \mathsf{C}, \mathsf{E}, \mathsf{Y} \rangle$ ends when Y requests deletion from C. The data subject's delete message activates the controller, who can then remove Y's data. The ideal execution $\langle \mathsf{C}(R'_\mathsf{C}), \mathsf{D}(\vec{\mathsf{q}}_\mathsf{E}) \rangle$ ends after D sends its last query from $\vec{\mathsf{q}}_\mathsf{E}$ to C. In both cases, the execution ends after the final activation of C. We consider the controller's state at the end of the execution: $\mathsf{state}_\mathsf{C} = \mathsf{C}(\vec{\mathsf{q}}_\mathsf{E}; R_\mathsf{C})$ in the real world, and $\mathsf{state}'_\mathsf{C} = \mathsf{C}(\vec{\mathsf{q}}_\mathsf{E}; R'_\mathsf{C})$ in the ideal world. If an execution never ends, the state is defined to be $\perp$ and the transcript τ and its subset $\vec{\mathsf{q}}_\mathsf{E}$ are defined to be empty. For example, the real execution ends if and only if Y requests deletion.

Remark 2.2 (Keeping time)**.** *In Section 5, we need a global clock. Balancing modelling simplicity with generality, we allow the environment to control time. Specifically, we introduce a special query* tick *that* E *can send to* C *thereby incrementing the clock. We do not allow* Y *to query* tick.

2.3 Defining Deletion-as-Control

We require that the internal state of the controller is about as likely in the real world and the ideal world, where probability is taken over C's random coins. Let $\mathsf{state}_\mathsf{C}$ and $\mathsf{state}'_\mathsf{C}$ be the internal states in the real and ideal executions. Consider a random variable R'_C which is sampled uniformly conditioned on $\mathsf{state}'_\mathsf{C} = \mathsf{state}_\mathsf{C}$ in the ideal execution where C uses randomness R'_C. Informally, our definition requires that the distributions of R'_C and R_C are close: $R'_\mathsf{C} \overset{\epsilon,\delta}{\approx} R_\mathsf{C}$. We do not require that the real and ideal executions are themselves (ϵ, δ)-indistinguishable. Instead, we require that the "explanations" in the real and ideal executions are (ϵ, δ)-indistinguishable—viewing the controller's randomness as the explanation of its state (relative to the environment's queries $\vec{\mathsf{q}}_\mathsf{E}$).

We extend this idea by considering ways of sampling R'_C other than the conditional distribution described above (which may not always be defined). In general, we allow a simulator Sim to sample R'_C as a function of the queries $\vec{\mathsf{q}}_\mathsf{E}$ from E to C, the real-world randomness R_C, and the real-world state $\mathsf{state}_\mathsf{C}$. (Although we view the simulator's output as an infinite-length bit sequence, it actually only needs to output a finite prefix.) We require that $R'_\mathsf{C} \overset{\epsilon,\delta}{\approx} R_\mathsf{C}$ and that $\mathsf{state}'_\mathsf{C} = \mathsf{state}_\mathsf{C}$ (or $\mathsf{state}_\mathsf{C} = \perp$) except with probability δ. Sampling R'_C conditioned on $\mathsf{state}'_\mathsf{C} = \mathsf{state}_\mathsf{C}$ is a useful default simulation strategy that we use throughout the paper, but there are sometimes much simpler ways to sample R'_C.

Definition 2.3 (Deletion as control)**.** *Given a controller* C*, an environment* E*, a data subject* Y *and a simulator* Sim*, we consider the following experiment:*

- $R_\mathsf{C} \leftarrow \mathcal{U}$ *(R_C is a uniform random tape)*
- $(\tau, \mathsf{state}_\mathsf{C}) \leftarrow \langle \mathsf{C}(R_\mathsf{C}), E, \mathsf{Y} \rangle$*, where* $\vec{\mathsf{q}}_\mathsf{E} \subseteq \tau$ *are the messages from* E *to* C*.*
- $R'_\mathsf{C} \leftarrow \mathsf{Sim}(\vec{\mathsf{q}}_\mathsf{E}, R_\mathsf{C}, \mathsf{state}_\mathsf{C})$
- $(\tau', \mathsf{state}'_\mathsf{C}) \leftarrow \langle \mathsf{C}(R'_\mathsf{C}), \mathsf{D}(\vec{\mathsf{q}}_\mathsf{E}) \rangle$.

We say a controller C *is* (ϵ, δ)-deletion-as-control compliant *if there exists* Sim *such that for all* E *and* Y *and for* R'_C, R_C *sampled as above:*

(1) $R'_\mathsf{C} \overset{\epsilon,\delta}{\approx} R_\mathsf{C}$ *(i.e.,* R_C *and* R'_C *are similarly distributed), and*
(2) With probability at least $1 - \delta$*, either* $\mathsf{state}'_\mathsf{C} = \mathsf{state}_\mathsf{C}$ *or* $\mathsf{state}_\mathsf{C} = \perp$.

For a particular class of data subjects $\mathcal{Y}$*, we say a controller* C *is* (ϵ, δ)-deletion-as-control compliant for $\mathcal{Y}$ *if there exists* Sim *such that the above holds for all* E *and for all* $\mathsf{Y} \in \mathcal{Y}$.

Example 2.4 (XOR Controller)**.** *Consider a controller* $\mathsf{C}_\oplus$ *which maintains a* k*-bit state* $\mathsf{state} \in \{0,1\}^k$ *which is initialized uniformly at random. Upon receiving a message* $x \in \{0,1\}^k$*,* $\mathsf{C}_\oplus$ *updates its state* $\mathsf{state} \leftarrow \mathsf{state} \oplus x$*, sending nothing in return. If it receives any other message, including* delete*, it does nothing.*

$\mathsf{C}_\oplus$ *satisfies* $(0,0)$*-deletion-as-control. To see why, consider an execution of the real world, which ends when the data subject sends* delete*. At this point,* $\mathsf{state} = R \oplus x_\mathsf{Y} \oplus x_\mathsf{E}$*, where* $R \in \{0,1\}^k$ *is the random initialization,* x_Y *is the XOR of all messages* x *sent by* Y*, and* x_E *is the XOR of all messages* x *sent by* E*. Let* Sim *compute* x_E *from the queries* $\vec{\mathsf{q}}_\mathsf{E}$*, and output* $R' = \mathsf{state} \oplus x_\mathsf{E} = R \oplus x_\mathsf{Y}$*. This satisfies the definition: (1)* R' *is uniformly distributed because* x_Y *is independent of* R*; (2)* $\mathsf{state}' = R' \oplus x_\mathsf{E} = \mathsf{state}$.

Definition 2.5 (Default simulator)**.** *The default simulator* Sim^* *samples* R'_C *as follows:*

$\mathsf{Sim}(\vec{\mathsf{q}}_\mathsf{E}, R_\mathsf{C}, \mathsf{state}_\mathsf{C})$:
Return $R'_\mathsf{C} \sim \mathcal{U}\big|_{\mathsf{C}(\vec{\mathsf{q}}_\mathsf{E}; R') = \mathsf{state}_\mathsf{C}}$ *if such an* R' *exists;*
Otherwise, return $R'_\mathsf{C} = R_\mathsf{C}$.

The latter case can occur if Y never requests deletion but $\vec{\mathsf{q}}_\mathsf{E}$ is finite, for example. The conditional distribution is well defined, since if there is an R' which leads to $\mathsf{C}(\vec{\mathsf{q}}_\mathsf{E}; R') = \mathsf{state}_\mathsf{C}$, then the event must occur with positive probability (because the execution of C terminated, it uses only a finite prefix of R'). In the full version, we prove that E and Y can be assumed to be deterministic without loss of generality.

2.4 Discussion of the definition

On constraining C*'s state.* Our definition imposes a condition on the internal state of the controller at a moment in time. Namely that, immediately after the data subject Y is deleted, the actual state of the controller $\mathsf{state}_\mathsf{C}$ can be plausibly attributed to the interaction between the controller and the environment alone. This in turn provides a guarantee for anything the controller may do in the future. Namely, if the real controller was replaced by the ideal

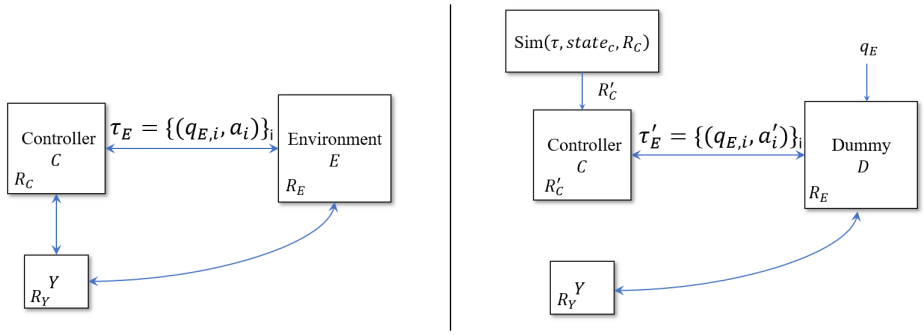

Figure 3: The real (left) and ideal (right) executions. The communications with the controller C are meant to be "public API" calls (e.g. accessing a public website, possibly using login credentials). Notice that the dummy environment in the ideal world sends message $\vec{q}_{E,i}$ regardless of what the previous responses were.

controller at the moment of Y's deletion, the environment would never know.

As an alternative, one might consider restricting future behavior directly. We prefer to restrict the state as well. It is simpler to describe—for instance, because there is a natural termination condition. It is also future-proof: Any controller that satisfied the future-behavior version but not the state version could choose to, in the future, violate the guarantee by publishing the state at the time of Y's erasure. Imposing the condition on the state directly makes it impossible for the future behavior of the real and ideal controllers to deviate, rather than merely possible for them not to.

Differential privacy and deletion. As we will see later in this paper, differential privacy (DP) can in some cases provide deletion-as-control almost automatically, with no additional action required of the data controller (Prop. 5.1 and Thm. 5.3). We believe that this makes sense both from the point of view of what DP means and the spirit of data protection regulations. When greater protection is warranted, deletion-as-control should not serve as the sole basis of analysis—nor should, perhaps, a right to erasure.

We're guided by a simple intuition: If a single individual has almost no influence on the result of data processing—the condition guaranteed by differential privacy—then nothing needs to be done to remove that individual's influence. This intuition closely tracks some prior approaches to deletion. For instance, to show that differentially private controllers satisfy deletion-as-*control*, we actually show that they meet the much stricter requirements of (approximate) deletion-as-*confidentiality* [15]. Some existing machine unlearning algorithms embody the same intuition, leaving the trained model unaltered as long as the deleted data points had no effect on the resulting model [16].[6]

The fact that DP can provide deletion-as-control fits well with the data protection regulations, like GDPR and CCPA, that inspire our work. Generally, these laws give individuals rights regarding the processing of *personal data* relating to them.But these rights, including the right to erasure, do not extend to data that have been sufficiently anonymized.[7] If one believes that in some cases, DP anonymizes data for the purposes of GDPR, say, then in such cases the data controller need not take any further action when a data subject requests deletion. Whether DP releases constitute personal data is explored in recent work bridging computer science formalisms with legal analysis [2, 34]. Though the general question remains unresolved, DP has been used to argue compliance with privacy laws for several high-profile data releases, including by the U.S. Census Bureau [40], Facebook [28], and Google [21].

Of course, DP is not always the answer. For example, if a model was trained using data collected without proper consent, one might require that no benefit derived from the ill-gotten data remains. The Federal Trade Commission first adopted this type of *algorithmic disgorgement* in a 2021 settlement with photo sharing app Everalbum [37]. Differential privacy should not shield against such algorithmic disgorgement.[8]

Deleting groups. Our definition provides a guarantee for an individual data subject. What about groups? If many people request to be deleted, then each individual person enjoys the individual-level guarantee provided by deletion-as-control. But the group does not necessarily enjoy an analogous group-level guarantee. For example, the group-level deletion guarantee for the DP-based controllers in Section 5 decays linearly with the group size. For large groups, the group *as a group* doesn't enjoy meaningful protection.

This seems unavoidable in contexts where (useful) statistics are published once and not subsequently updated. It reflects a fundamental difference between deletion-as-control and the history independence-style definitions in the machine unlearning literature, discussed in Section 3.4. Suppose, for example, that a controller trains a model θ using data from n people. Then all n people request deletion, leaving the controller with a model θ^*. History independence would require that θ^* be essentially trivial: θ^* should perform

[6] In contrast, Thudi et al. [38] argue that any definition where a data controller "do[es] not need to do anything and can claim the unlearning is done"—including machine unlearning definitions based on approximate history independence (Section 3.4)—is "not well-defined". We disagree.

[7] Recital 26 states this explicitly: "The principles of data protection should therefore not apply to anonymous information, namely information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable."

[8] Achille et al. [1] seem to disagree, writing: "In many ways, differential privacy (DP) can be considered the 'gold standard' of model disgorgement".

about as well as a the model θ_0 trained on an empty dataset (Section 3.4). On the other hand, the DP-based controllers in Section 5 allow θ^* to perform very well on the learning task.

An individual-level guarantee is in line with data privacy laws. To whit, the GDPR grants a right to erasure to "the data subject" who is "[a] natural person" (Art. 4, 17). Even so, group-level deletion may be more appropriate in some settings (e.g., algorithmic disgorgement discussed above). Exploring group deletion is an important direction for future work.

Composition. Composition is an important property of good cryptographic definitions. We do not yet have a complete picture of how deletion-as-control composes. Theorem 6.4 states a limited composition theorem that applies to parallel composition of two controllers at least one of which satisfies a very strong guarantee (specifically, it must implement a deterministic functionality with perfect, as opposed to approximate, deletion-as-control). By induction, this extends to the parallel composition of k controllers if all but one satisfy the strong guarantee. This can be used to reason about complex interactive functionalities built from multiple strongly history-independent data structures.

Proving more general composition for deletion-as-control is an important question for future work. Addressing it seems challenging since it is closely related to still-open questions about composition for differential privacy. For example, it was shown only very recently that differential privacy composes when mechanisms are run concurrently with adaptively interleaved queries [41]. While that result allows adaptive query ordering, the dataset itself is fixed in advance. Deletion-as-control allows both queries and data to be specified adaptively. Proving composition of deletion-as-control seems only harder than the analogous question for differential privacy.

Other limitations of our approach. We touch on two limitations of our approach. First, there is no quantification of "effort." The EU's right to be forgotten stems from *Google v Costeja*, where the Court of Justice for the European Union ruled that a search engine may be required to remove certain links from search results [11]. But there are limits. Today, Google will only remove the result from search queries related to the name of the person requesting deletion, but not from other search queries [20]. This suggests a definition in which results are hidden from a low-resource adversary who only makes general searches, but not from an adversary with more side information or time, carrying out more targeted or exhaustive searches respectively. Modeling that sort of subtlety appears to require fundamental changes from all existing approaches, ours included.

Second, a failure of deletion as we formulate it doesn't map to an explicit attack on a system. It corresponds instead to a disconnect between the real execution and a counterfactual one in which Alice's data never existed but her effect on others' data remains. In this sense the definition is quite different from standard cryptographic ones, and it doesn't obviously correspond to an adversarial model nor combine well with other cryptographic definitions. This is also true of the history-independence approach, including the definitions in prior work on machine unlearning. Deletion-as-confidentiality [15] does have a more straightforward cryptographic flavor but, as we argue, its strict requirement is ill suited for many application.

3 HISTORY INDEPENDENCE AND DELETION-AS-CONTROL

History independence (HI) is concerned with the problem that the memory representation of a data structure may reveal information about the history of operations that were performed on it [24, 29, 30]. HI requires that the memory representation reveals nothing more than the current logical state of the data structure.

In this section, we state the definition of (non-adaptive) history independence (Section 3.1). We then define a more general notion that allows for implementations that satisfy the conditions of HI *approximately* and *adaptively* (Section 3.2). The generalization is complex since we must explicitly model adaptivity in the interactions between a data structure and those issuing queries to it. Briefly, an adaptive adversary A interacting with the data structure produces two equivalent query sequences. Adaptive history independence (AHI) requires that the joint distribution of A's view and the data structure's state is the same under both sequences. Approximate AHI requires these distributions to be (ϵ, δ)-close.

We show that data controllers that satisfy approximate AHI also satisfy our notion of deletion-as-control (Section 3.3) with the same parameters. Finally, we show how existing definitions of *machine unlearning* and the corresponding constructions are all (weakenings of) our general notion of history independence (Section 3.4).

3.1 History independence

An *abstract data type (ADT)* is defined by a universe of operations $\{\mathrm{op}\}$ and a mapping $\mathrm{ADT} : (\mathrm{op}, s_{\mathrm{adt}}) \mapsto (s'_{\mathrm{adt}}, \mathrm{out}_{\mathrm{adt}})$. We call $s_{\mathrm{adt}}, s'_{\mathrm{adt}} \in \{0,1\}^*$ the *logical states* before and after operation, where s'_{adt} is a deterministic function of s_{adt} and op. We call $\mathrm{out}_{\mathrm{adt}} \in \{0,1\}^* \cup \{\perp\}$ the *logical output*, which may be randomized. In subsequent sections we will assume without loss of generality that operations $\mathrm{op}(id)$ are tagged by $id \in \{0,1\}^*$. We omit the tags where possible to reduce clutter.

Given an initial logical state s^0_{adt} and a sequence of operations $\sigma = (\mathrm{op}^1, \mathrm{op}^2, \dots)$, the ADT defines a sequence of logical states $(s^1_{\mathrm{adt}}, s^2_{\mathrm{adt}}, \dots)$ and a sequence of outputs $\overrightarrow{\mathrm{out}} = (\mathrm{out}^1_{\mathrm{adt}}, \mathrm{out}^2_{\mathrm{adt}}, \dots)$ by iterated application of ADT. We denote by $\mathrm{ADT}(\sigma).\mathrm{state}$ the final logical state that results from this iterated application. When no initial state is specified, it is assumed to be the empty state.

Definition 3.1. *We say two sequences of operations σ and σ' are* logically equivalent, *denoted $\sigma \overset{L}{\equiv} \sigma'$, if $\mathrm{ADT}(\sigma).\mathrm{state} = \mathrm{ADT}(\sigma').\mathrm{state}$. Logical equivalence is an equivalence relation, and we denote by $[\sigma]$ a canonical sequence in the equivalence class of sequences $\{\sigma' : \sigma' \overset{L}{\equiv} \sigma\}$.*

An *implementation* (e.g., a computer program for a particular architecture) is a possibly randomized mapping $\mathsf{Impl} : (\mathrm{op}, s) \mapsto (s', \mathrm{out})$. We call s the *physical state* and out the *physical output*. Both may be randomized. Given an initial state and sequence of operations, Impl defines a sequence of physical states and outputs by iterated application. When no initial state is specified, it is assumed to be the empty state.

Definition 3.2 (History independence [30])**.** Impl *is a* weakly history independent *implementation of* ADT (WHI-implements ADT) *if*

$$\sigma \overset{L}{\equiv} \sigma' \implies \mathsf{Impl}(\sigma).\mathsf{state} \equiv \mathsf{Impl}(\sigma').\mathsf{state}, \tag{1}$$

where $\equiv$ *denotes equality of distributions.* Impl *is a* strongly history independent *implementation of* ADT *(SHI-implements ADT) if for all initial states* s

$$\sigma \overset{L}{\equiv} \sigma' \implies \mathsf{Impl}(\sigma, s).\mathsf{state} \equiv \mathsf{Impl}(\sigma', s).\mathsf{state}. \tag{2}$$

One can obtain approximate, *nonadaptive* versions of history independence by replacing $\equiv$ with $\overset{\epsilon,\delta}{\approx}$ in Definition 3.2. However, because the sequence of queries is specified ahead of time, such a definition's guarantees are not meaningful in interactive settings.

3.1.1 Strongly History Independent Dictionaries. To illustrate history independence, consider the dictionary ADT, which models a simple key-value store. Looking ahead, we will use history independent dictionaries to build deletion-compliant controllers from differential privacy. For our purposes, the keys will be party IDs id; the values can be arbitrary. The ADT supports operations $\mathsf{insert}(id)$, $\mathsf{delete}(id)$, $\mathsf{get}(id)$, and $\mathsf{set}(id, value)$, where set associates the key corresponding to id with $value$, and get returns the most recently set value. We assume that $\mathsf{insert}(id)$ is equivalent to $\mathsf{set}(id, \top)$ where $\top$ is a special default value.

Dictionaries are typically implemented as hash tables, but such data structures are generally *not* history independent.

Storing a dictionary as a sorted list is inefficient—updates generally take time $\Omega(n)$, where n is the current number of keys—but it enjoys strong history independence. Since we do not focus on efficiency here, the reader may think of the sorted list as our default implementation of a dictionary. [9]

3.2 Adaptive History Independence (AHI)

The history independence literature gives no guarantees against adaptively-chosen sequences of queries, because the two sequences σ and σ' are fixed before the implementation's randomness is sampled.

Inspired by [23], we extend the well-studied notion of history independence to the *adaptive* setting, where the sequence σ of operations is chosen adaptively by an algorithm interacting with an implementation of an ADT.

We consider an interaction $\langle \mathsf{Impl}(R), \mathsf{A} \rangle$ between an algorithm A and the implementation Impl with random tape $R \sim \mathcal{U}$.

In the interaction, A adaptively outputs an operation $\mathsf{op}^i \leftarrow \mathsf{A}(\mathsf{op}^1, \mathsf{out}^1, \ldots, \mathsf{op}^{i-1}, \mathsf{out}^{i-1})$, and receives the output out^i in return. The interaction defines a sequence of operations σ and corresponding outputs $\overrightarrow{\mathsf{out}}$. Eventually, A outputs a sequence σ^* that is logically equivalent to the sequence σ of operations performed so far. Impl is executed on σ^* and alternate randomness R^*, resulting in $s^* = \mathsf{Impl}(\sigma^*; R^*).\mathsf{state}$. We consider two variants: $R^* = R$, or $R^* \sim \mathcal{U}$ independent of R.

We consider the adversary's ability to distinguish the real state $s = \mathsf{Impl}(\sigma; R).\mathsf{state}$ and the logically equivalent state $s^* = \mathsf{Impl}(\sigma^*; R^*).\mathsf{state}$, given its view $V_{\mathsf{Impl,A}} = (\sigma, \sigma^*, \overrightarrow{\mathsf{out}})$. Our definition of adaptive history independence requires that the joint distributions of $(V_{\mathsf{Impl,A}}, s)$ and $(V_{\mathsf{Impl,A}}, s^*)$ be (ϵ, δ)-close. We restrict ourselves to adversaries A such that $\langle \mathsf{Impl}, \mathsf{A} \rangle$ always terminates, which we call *valid* adversaries.

Definition 3.3 (Adaptive (weak) history independence)**.** *An implementation* Impl *of an* ADT *is* (ϵ, δ)*-history independent (AHI) if for all valid adversaries* A*:*

$$(V_{\mathsf{Impl,A}}, s) \overset{\epsilon,\delta}{\approx} (V_{\mathsf{Impl,A}}, s^*)$$

where the distributions are given by the probability experiment described above.

Essentially all $(0, 0)$-strongly HI data structures can be shown to satisfy $(0, 0)$-AHI. The argument uses the following strong property of SHI [24]: For every setting of the random string $R = r$, and for every two logically equivalent sequences σ, σ^*, the data structure stores the same state $s(\sigma, r) = s(\sigma^*, r)$.

3.3 AHI and Deletion-as-Control

In Theorem 3.6, we state that data controllers that implement history independent ADTs satisfy deletion-as-control. This relationship to deletion-as-control only makes sense if the ADT itself supports some notion of deletion, which we define as follows.

Consider an ADT where operations $\{\mathsf{op}(id)\}$ are tagged with an identifier $id \in \{0, 1\}^*$ (e.g., the channel IDs). For a sequence σ of operations and an id^*, let σ_{-id^*} be the sequence of operations with every operation with identifier id^* removed (that is, $\mathsf{op}(id^*)$ for all values of op).

Definition 3.4 (Logical Deletion)**.** *ADT supports* logical deletion *if there exists an operation* delete *such that for all sequences of operations* σ *and for all IDs* id^**:* $\left(\sigma \| \mathsf{delete}(id^*)\right) \overset{L}{\equiv} \sigma_{-id^*}$.

Next, we define the following syntax to allow ADTs to interface with the deletion-as-control execution.

Definition 3.5 (Controller relative to an implementation)**.** *Let* Impl *be an implementation of an* ADT*. We define the controller* $\mathsf{C}_{\mathsf{Impl}}$ *relative to* Impl *as the controller that maintains state* state *and works as follows:*

- *On input* $(\mathsf{cID}, \mathsf{msg})$*:*
  - $(\mathsf{state}', \mathsf{out}) \leftarrow \mathsf{Impl}(\mathsf{op}(\mathsf{cID}), \mathsf{state})$*, where* $\mathsf{op} \leftarrow \mathsf{msg}$.
  - *Write* $(\mathsf{cID}, \mathsf{out})$ *to the output tape.*
- *On input* fail*: Halt.*

Theorem 3.6. *For any ADT that supports logical deletion and any* Impl *of the ADT satisfying* (ϵ, δ)*-AHI (with either variant of Definition 3.3), the controller* $\mathsf{C} = \mathsf{C}_{\mathsf{Impl}}$ *is* (ϵ, δ)*-deletion-as-control compliant.*

The proof in the full version [10] uses a simple, novel result on indistinguishability, dubbed the Coupling Lemma, which we present next.

Lemma 3.7 (Coupling Lemma)**.** *Let* P, Q *be probability distributions on sets* $\mathcal{X}$ *and* $\mathcal{Y}$*, respectively, and let* $f : \mathcal{X} \to \mathcal{Z}$ *and* $g : \mathcal{Y} \to \mathcal{Z}$ *be (deterministic) functions with the same codomain* $\mathcal{Z}$*. Suppose that* $f(X) \approx_{\epsilon,\delta} g(Y)$ *when* $X \sim P$ *and* $Y \sim Q$*.*

[9] In fact, a strongly history independent hash table implementation with constant expected-time operations was described by Blelloch and Golovin [4].

Consider the collection of distributions $\{Q_x \in \Delta(\mathcal{Y}) : x \in \mathcal{X}\}$ *on the set* $\mathcal{Y}$*, where* Q_x *denotes the distribution on* Y *conditioned on the event that* $g(Y) = f(x)$ *(that is* $Q_x = Q|_{\{y:g(y)=f(x)\}}$*). In the case that* $g^{-1}(f(x))$ *is empty,* Q_x *assigns probability to a default value* $\perp \in \mathcal{Y}$*.*

If we select $X \sim P$ *and then sample* $Y' \sim Q_X$ *(so that* $\Pr(Y' = y|X = x) = Q_x(y)$*), then*

(1) $Y' \approx_{\epsilon,\delta} Y$*, and*
(2) $f(X) = g(Y')$ *with probability at least* $1 - \delta$ *over* (X, Y)*.*

As a corollary to Theorem 3.6, any history independent implementation of the Private Cloud Storage touchstone or the Public Bulletin Board examples (from Section 1.1) satisfy deletion-as-control.

The Private Cloud Storage functionality works like a key-value store: each user has their own dictionary D_{id}. To upload a file, a user sends $Upload(id, filename, file)$ to the controller, which internally adds $(filename, file)$ to D_{id} (if $filename$ already exists in D_{id} then nothing happens). To download one of their own files, the user sends $Download(id, filename)$ to the controller and receives the corresponding $file$ from D_{id}, if one exists. To delete their account, the user sends $Delete(id)$, in which the controller removes dictionary D_{id}. Two sequences of operations are logically equivalent if, for every user id, the dictionary D_{id} has the same logical content after applying the two sequences.

We define the Public Bulletin Board functionality as follows: users can post a message by sending $Post(id, msg)$ to the controller; they can receive all messages currently on the board by sending $Read()$; and they can delete all of their messages by sending $Delete(id)$ to the controller. Internally, the Public Bulletin Board stores an ordered list of (id, msg) pairs (ordered by insertion time). Two sequences of operations are logically equivalent if they yield the same ordered list of (id, msg) pairs.

Corollary 3.8. *The Private Cloud Storage and Public Bulletin Board touchstone controllers (Section 1.1) implemented with* (ϵ, δ)*-adaptive history independence each satisfy* (ϵ, δ)*-deletion-as-control.*

3.4 From Prior Definitions of Machine Unlearning to History Independence

We claim that AHI captures the essence of existing definitions of "machine unlearning" (that is, protocols that update a machine learning model to reflect deletions from the training data). Each definition in the literature corresponds to a special case of history independence, though each weakens the definition in one or more ways (even when their constructions satisfy the stronger, general notion). For illustration, we discuss the approach of Gupta et al. [23] in detail.

The basic correspondence comes via considering an abstract data type, which we dub the *Updatable ML*, that extends a dictionary: it maintains a multiset $\vec{x}$ of labeled examples from some universe $\mathcal{Z}$. In addition to allowing insert() and delete() operations, it accepts a possibly randomized operation predict which outputs a predictor (or other trained model) ψ trained on $\vec{x}$. The accuracy requirement for ψ is generally not fully specified, not least because many current machine learning methods don't come with worst-case guarantees. The literature on machine unlearning generally requires that the distribution of the final predictor ψ (e.g., the model parameters) is approximately the same as it would be for a minimal sequence of operations that leads to the same training data set. In particular, *deleting an individual* Y *should mean that* ψ *looks roughly the same as if* Y *had never appeared in* $\vec{x}$. In principle one could satisfy the requirement by simply retraining ψ from scratch every time the data set changes, though this may be practically infeasible. The literature therefore focuses on methods that allow for faster updates.

Specializing history independence to updatable ML. First, we spell out how AHI (Definition 3.3) specializes to the Updatable ML ADT. Given a sequence of updates and prediction queries $\sigma = (u_1, ..., u_t)$, let $[\sigma]$ denote a canonical equivalent sequence of updates. For example, $[\sigma]$ may (a) discard any insertion/deletion pairs that operate on the same element, with the insertion appearing first, and (b) list the remaining operations in lexicographic order. In an interaction with an implementation Impl that is initialized with an empty data set, an adversary makes a sequence of queries σ (of which only the updates actually affect the state) and, eventually, terminates at some time step t. Let s the resulting state of the controller based on randomness R, and let $\vec{x}_t$ be the final logical data set. After the interaction ends, the adversary outputs some logically equivalent sequence σ^* (for example, it could choose the canonical sequence $\sigma^* = [\sigma]$). Finally, run Impl from scratch with randomness R' and queries σ^* to get state s^*. The logical data set $\vec{x}_t$ is the same for σ and σ^* by construction. Definition 3.3 requires that

$$(\sigma, s) \overset{\epsilon,\delta}{\approx} (\sigma, s^*)\,. \tag{3}$$

Definitions from the literature. The exact definition of deletion varies from paper to paper (and some papers do not define terms precisely). All formulate the problem in terms of what we call the Updatable ML ADT. We claim their definitions are variations on adaptive history independence, each with one or more of the following weakenings:

Restricted queries Some papers consider only a subset of allowed operations—e.g., in Ginart et al. [16], Sekhari et al. [36], Ullah et al. [39] data is inserted as a batch, and then only deletions occur.

One output versus future behavior Some consider only the output of the system at one time (e.g., Gupta et al. [23], Neel et al. [32], Sekhari et al. [36]), while others also consider the internal state (e.g., Ginart et al. [16]). The narrower approach *does not constrain the future behavior of the system.* Among those definitions that consider the full state, none discuss issues of internal representations; in this section, we also elide this distinction, assuming that datasets can be represented internally using strongly history-independent data structures.

Nonadaptive queries Except for Gupta et al. [23], the literature considers only adversaries that specify the set of queries to be issued in advance. For constructions that are $(0, 0)$-HI (that is, in which the real and ideal distributions are identical), this comes at no loss of generality. However, the nonadaptive and adaptive versions of (ϵ, δ)-HI for $\delta > 0$ are very different [23]. Even in Gupta et al. [23], the length t of the query sequence is chosen nonadaptively.

Symmetric vs asymmetric indistinguishability Ginart et al. [16] and [23] consider a one-sided weakening of (ϵ, δ)-indistinguishability.

The definitions of Bourtoule et al. [5], Ginart et al. [16], Ullah et al. [39] consider *exact* variants of unlearning (with $\epsilon = \delta = 0$), while others Golatkar et al. [18, 19], Guo et al. [22], Gupta et al. [23], Neel et al. [32], Sekhari et al. [36] consider approximate variants.

Despite generally formulating weaker definitions, the algorithms in the literature often satisfy history independence. In particular, because the constructions in Cao and Yang [6], Ginart et al. [16] are fully deterministic, it is easy to see that they satisfy adaptive, $(0, 0)$-*strong history independence* (by Theorem 1 in [24]). Some constructions satisfy additional properties, such as storing much less information than the full data set $\vec{x}_t$ (e.g., Cao and Yang [6], Sekhari et al. [36]).

We discuss the relationship to one previous definition—that of Gupta et al. [23]—in more detail in the full version, since the definition is subtle and the relationship is technically nontrivial. Briefly: their definition is similar to AHI for the Updatable ML ADT, but is weaker in that it looks at only one output (not all future behavior), and that it uses a one-sided version of the indistinguishability condition. We conjecture that the specific algorithms in Gupta et al. [23] satisfy our stronger condition.

4 DELETION-AS-CONFIDENTIALITY AND DELETION-AS-CONTROL

We study the relationship between the notion of deletion as confidentiality from Garg et al. [15] (hereafter "the GGV definition") and our notion of deletion-as-control (Definition 2.3). Similar to Definition 2.3, deletion-as-confidentiality also considers two executions, real and ideal. At the end of the two executions, they compare the *view* of the environment V_{E} and the state of the controller $\text{state}_{\mathsf{C}}$. Simplifying away (important but technical) details, the real GGV execution is roughly the same as in deletion-as-control. The ideal GGV execution simply drops all messages between Y and C. Informally, deletion-as-confidentiality requires that E's view and C's final state are indistinguishable in the real and ideal worlds.

In Section 4.1, we define *deletion-as-confidentiality*—the definition closest in spirit to the GGV definition but in our execution model. (We cannot directly compare our definition to GGV as they are defined in different models of interaction, stemming from a need in deletion-as-control to 'sync' the real and ideal executions.) In Section 4.2, we show that for many data subjects Y, deletion-as-confidentiality for Y implies deletion-as-control for Y (Theorem 4.3). Finally, we show that the implication cannot hold for all data subjects due to the differences in the execution models (Theorems 4.4 and 4.5).

4.1 Definition

Below, we define deletion-as-confidentiality by describing how it differs from deletion-as-control. After the definition, we briefly explain how our adaptation of deletion-as-confidentiality differs from that in [15]. Similar to Definition 2.3, deletion-as-confidentiality also considers two executions—real and ideal. The real execution is involves the three parties C, E, and Y. It defines the *view* of the environment E, denoted V_{E}^{real} and the state of the controller C, denoted $\text{state}_{\mathsf{C}}^{real}$, at the end of the execution. V_{E}^{real} includes E's randomness and transcript $\tau_{\mathsf{E}}^{real} = (\vec{\mathsf{q}}_{\mathsf{E}}^{real}, \vec{\mathsf{a}}_{\mathsf{E}}^{real})$. The GGV definition requires that $(\text{state}_{\mathsf{C}}^{real}, V_{\mathsf{E}}^{real})$ be indistinguishable from the state and view in the ideal world where the data subject never communicates with anybody. This definition inherently requires that C can never reveal *any* information about one user's data or participation to another data or participation to another. GGV and deletion-as-control differ mainly in 3 ways:

Ideal execution The ideal GGV execution also involves the same three parties C, E, and Y—no dummy party D. The ideal GGV execution drops all messages between Y and C. This execution results in some view V_{E}^{ideal} and some state $\text{state}_{\mathsf{C}}^{ideal}$.

Termination The real deletion-as-control execution ends after C processes Y's first delete message. The GGV execution ends when E sends a special `finish` message to C. (Y then sends delete if it hasn't already.) Hence, the end time can depend on Y's view in deletion-as-control, but not in GGV. This difference reflects our two fundamentally different approaches to defining the ideal execution. (Thm 4.5 leverages this gap.)

Indistinguishability GGV requires (ϵ, δ)-indistinguishability of both V_{E} and $\text{state}_{\mathsf{C}}$. In contrast, deletion-as-control imposes no requirement on V_{E}. Moreover, GGV only requires indistinguishability for Y's that don't send any messages to E.[10]

Definition 4.1 ((ϵ, δ) Deletion-as-Confidentiality for Y)**.** *For a data subject* Y*, a controller* C *is* (ϵ, δ)-deletion-as-confidentiality compliant for Y *if for all* E*, in the executions involving* C, E, Y,

$$(V_{\mathsf{E}}^{real}, \text{state}_{\mathsf{C}}^{real}) \overset{\epsilon,\delta}{\approx} (V_{\mathsf{E}}^{ideal}, \text{state}_{\mathsf{C}}^{ideal}).$$

Definition 4.2 (Deletion-as-Confidentiality, adapted from [15])**.** *Let* $\mathcal{Y}_{\text{silent}}$ *be the set of data subjects* Y *that never send any messages to* E. C *is* (ϵ, δ)-deletion-as-confidentiality compliant *if it is* (ϵ, δ)-*deletion-as-confidentiality compliant for all* $\mathsf{Y} \in \mathcal{Y}_{\text{silent}}$.

The version of deletion-as-confidentiality in Definition 4.2 differs from the original definition of Garg et al. [15] in a few important ways. First, [15] allows users to request deletion of the information shared in specific interactions between Y and C. Simplifying, we take deletion of a user's data to be all or nothing. Second, [15] allows users to delete many times, whereas we focus on a single deletion. Third, C's randomness tape is not read-once in [15]. Finally, the execution model of [15] does not have authenticated channels, which they show is necessary for non-trivial functionalities. In light of this, we chose to build authentication into our execution model.

4.2 Control vs. Confidentiality

In spirit, deletion-as-confidentiality imposes a stronger indistinguishability requirement than deletion-as-control. The former requires that no information about the deleted data is ever revealed, whereas the latter only requires that the effect of the deleted data is not present *after* the deletion happens. One might thus expect that any C that satisfies Definition 4.2 would also satisfy Definition 2.3. But deletion-as-control captures more general environments and

[10] Otherwise, GGV would rule out functionalities where Y can determine if its messages are delivered to C (e.g., getting an acknowledgement).

data subjects than deletion-as-confidentiality. First, deletion-as-control allows Y to communicate freely with E, whereas deletion-as-confidentiality does not (i.e., $\mathsf{Y} \in \mathcal{Y}_{\mathsf{silent}}$). Second, deletion-as-control imposes a requirement as soon as Y deletes, whereas deletion-as-confidentiality only requires indistinguishability after E terminates the execution (which may be much later).

Let $\mathcal{Y}_{\mathsf{silent}}$ be as in Definition 4.2 and $\mathcal{Y}_{\mathsf{dummy}}$ be the set of Y that only delete when instructed by E. Let $\mathcal{Y}_{\mathsf{lift}} = \mathcal{Y}_{\mathsf{silent}} \cap \mathcal{Y}_{\mathsf{dummy}}$.

Theorem 4.3. *For any* C *and any* $\mathsf{Y} \in \mathcal{Y}_{\mathsf{lift}}$*, if* C *is* (ϵ, δ)*-deletion-as-confidentiality compliant for* Y*, then it is also* (ϵ, δ)*-deletion-as-control compliant for* Y.

Theorems 4.4 and 4.5 show that the restriction in Theorem 4.3 that $\mathsf{Y} \in \mathcal{Y}_{\mathsf{lift}} = \mathcal{Y}_{\mathsf{dummy}} \cap \mathcal{Y}_{\mathsf{silent}}$ is necessary.

Theorem 4.4. *There exists* C *and* $\mathsf{Y} \in \mathcal{Y}_{\mathsf{dummy}} \setminus \mathcal{Y}_{\mathsf{silent}}$ *such that* C *satisfies* $(0, 0)$*-deletion-as-confidentiality, but* C *does not satisfy* (ϵ', δ')*-deletion-as-control for* Y *any* $\epsilon' < \infty$, $\delta' < 1$.

Theorem 4.5. *For any* $\delta > 0$*, there exists* C *and* $\mathsf{Y} \in \mathcal{Y}_{\mathsf{silent}}$ *such that (i)* C *satisfies* $(0, \delta)$*-deletion-as-confidentiality for* Y*, and (ii)* C *does not satisfy* (ϵ', δ')*-deletion-as-control for* Y *for any* $\epsilon < \infty$, $\delta' < 1$.

5 DIFFERENTIAL PRIVACY AND DELETION-AS-CONTROL

This section describes two ways of compiling (ϵ, δ)-differentially private mechanisms $\mathcal{M}$ into controllers satisfying (ϵ, δ)-deletion-as-compliance. The first applies to DP mechanisms that are run in a batch setting on a single, centralized dataset: data summarization, query release, or DP-SGD, for example (Section 5.1). The second applies to mechanisms satisfying pan-privacy under continual release (Section 5.2) . Along the way we define non-adaptive event-level pan privacy (one intrusion) with continual release, first defined by Chan et al. [8], Dwork et al. [13], and an adaptive variant, building on the adaptive continual release definition of Jain et al. [25]. Both of our compilers make use of a SHI dictionary $\mathcal{D}$ (Section 3.1.1).

There is a strong intuitive connection between (approximate) history independence and deletion-as-control, as illustrated by the results of the previous section. This intuition is so strong that many prior works on machine unlearning *essentially equate* deletion with history independence (Section 3.4).

However, *the examples of this section show that our notion of deletion-as-control is much broader than history independence.* For instance, consider a controller as follows. At some time t_0 the controller computes a differentially-private approximation out_{t_0} to the current number of users in the data set, and the controller stores out_{t_0} and makes it available at all later times. Intuitively, this controller does not satisfy any version of history independence: even if every user request deletion at time $t_0 + 1$, the stored out_{t_0} is unchanged—making this history easy to distinguish from one where the all users deleted at time $t_0 - 1$. It could, however, still satisfy deletion-as-control.

Keeping time. Throughout this section we consider systems with a global clock. This allows for controllers that publish the number of weekly active users, say. For simplicity and generality, we allow the environment to control time. Specifically, we introduce a special query tick that only E can send to C to increment the clock.

5.1 Batch differential privacy

Let $\mathcal{M} : \mathcal{D} \mapsto \mathcal{M}(\mathcal{D})$ be a non-interactive differentially private mechanism $\mathcal{M}$.

We briefly describe a simple controller $\mathsf{C}^{\mathsf{batch}}_{\mathcal{M}}$ that satisfies deletion-as-control. Detailed pseudo-code is in the full version [10].

It works in three phases: before tick, during tick, and after tick. At the beginning, $\mathsf{C}^{\mathsf{batch}}_{\mathcal{M}}$ populates a dataset $\mathcal{D}$ stored as a SHI dictionary from its input stream, returning $\perp$ in response to every query. When it receives the tick, it evaluates $\mathcal{M}(\mathcal{D})$, stores the result as out, and erases the dictionary $\mathcal{D}$. For all future queries, $\mathsf{C}^{\mathsf{batch}}_{\mathcal{M}}$ simply returns out. We assume for simplicity that the mechanism $\mathcal{M}$ is a function only of the logical contents of $\mathcal{D}$, and is independent of its memory representation.

Proposition 5.1.

(1) If $\mathcal{M}$ *is* (ϵ, δ)*-DP, then* $\mathsf{C}^{\mathsf{batch}}_{\mathcal{M}}$ *satisfies* (ϵ, δ)*-deletion-as-control.*

(2) For any $\epsilon > 0$*, suppose* $\mathcal{M}$ *is the Laplace mechanism with parameter* ϵ *applied to a count of the number of record in its input. Then* $\mathsf{C}^{\mathsf{batch}}_{\mathcal{M}}$ *satisfies* (ϵ, δ)*-deletion-as-control but is* not (ϵ', δ')*-HI for any* $\epsilon' < \infty$ *and* $\delta' < 1$.

The second part of the proposition really applies to any DP mechanism that releases useful information about its inputs—the argument relies just on the fact that $\mathcal{M}$ acts differently on the empty data set than it does on a data set with many records. In particular, the DP Machine learning example from Section 1.1, which trains a model using DP-SGD, satisfies deletion-as-control.

Corollary 5.2. *The* (ϵ, δ)*-DP Machine Learning touchstone controller (Section 1.1) satisfies* (ϵ, δ)*-deletion-as-control.*

5.2 Pan-privacy under continual release

In the full version [10], we formalize a definition of adaptive pan-privacy under continual release (against a single intrusion), extending the non-adaptive versions originally defined in [8, 13]. The continual release setting concerns an *online controller* that processes a stream of elements and produces outputs at regular time intervals. We prove the following theorem which gives a general transformation from an adaptive pan private mechanism $\mathcal{M}$ to a controller $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ satisfying deletion-as-control. Roughly, $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ emulates a copy of $\mathcal{M}$. For each *id*, $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ passes the first operation from *id* to $\mathcal{M}$ and filters out subsequent operations. It uses a SHI dictionary $\mathcal{D}$ to implement the filter. To delete, *id* is removed from $\mathcal{D}$ but $\mathcal{M}$ is unaffected. (Note that a deleted user can then issue a new query to $\mathcal{M}$; we make no guarantees for such users.)

Theorem 5.3 (Informal). *If a controller* $\mathcal{M}$ *satisfies* (ϵ, δ)*-adaptive event-level pan-privacy with continual-release, then* $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ *satisfies* (ϵ, δ)*-deletion-as-control.*

The proof first shows that $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ satisfies deletion-as-control, and then shows that—for $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ specifically—the restriction to $\mathcal{Y}_{\mathsf{lift}}$ in Theorem 4.3 is without loss of generality.

5.3 Examples of pan-private controllers

5.3.1 Counters. Perhaps the most widely studied algorithm that is pan-private under continual release is the tree mechanism of Chan et al. [8], Dwork et al. [12]. Although originally formulated without pan-privacy, it relies on the composition of summations over different time intervals. Each of these can be made pan-private against a single intrusion by adding noise when the counter is initialized and again at release [13]. The overall mechanism retains the same asymtptotic guarantees, since the noise at most doubles, and enjoys pan-privacy. We summarize its properties here:

Proposition 5.4 (Combining [13] with [8, 12])**.** *There is an (ϵ, δ)-adaptively event-level pan-private under continual-release algorithm that takes a time horizon T and a sequence of inputs $x_1, x_2, \ldots \in [0,1]$ and at the t-th* tick*, for $t \in [T]$, releases* out_t *such that*

$$\mathsf{out}_t = \sum_{\substack{i\,:\,x_i \text{ received} \\ \text{by time } t}} x_i + Z_t,$$

where Z_t's are distributed (independently of x_i's but not each other) as $Z_t \sim N(0, \sigma_t^2)$ with $\sigma_t = O\left(\frac{\sqrt{\log^3 T \log(1/\delta)}}{\epsilon}\right)$.

In Section 6, we use the tree mechanism as a building block to construct a controller that satisfies deletion-as-control but not history independence, confidentiality, nor differential privacy. Moreover, one can use the tree mechanism to construct a controller that satisfies deletion-as-control but not history independence (cf. Proposition 5.1). The controller could, for example, use the tree mechanism to count the number of distinct users n it has seen. Applying Theorem 5.3, a controller could publish $n + N(0, \sigma_t^2)$ even after all users have requested deletion, all while satisfying deletion-as-control. *Such functionality is impossible under history independence.*

5.3.2 Learning under Pan-Private Continual Release (and with Deletion). As an example, we show how the results of this section allow one to maintain a model trained via gradient descent with relatively little noise—the algorithm's error is similar to that of state of the art federated machine learning algorithms.

Consider the following optimization problem: given a loss function $\ell : \Theta \times \mathcal{X} \to \mathbb{R}$ and a data set $\vec{x} = (x_1, \ldots, x_n)$, we aim to find $h \in \Theta$ which approximately minimizes

$$\mathcal{L}_{\vec{x}}(h) = \sum_{i=1}^{n} \ell(h; x_i)\,. \tag{4}$$

To analyze convergence, we assume a convex loss function. Nevertheless, the method we give applies more broadly—the privacy or deletion guarantees hold regardless of convexity, and the algorithms makes sense as long as the loss function is roughly convex near its minimum.

Algorithms for this problem that are differentially private under continual release were studied for online learning and efficient distributed learning [27]. The algorithm they consider is not panprivate. However, it only access the data via the tree-based mechanism for continual release of a sum [8, 12] (in this case, releasing the sum of the gradients of users' contributions to the overall loss function over the evaluation of a first-order optimization algorithm). We observed above that this can be made panprivate with only a constant factor increase in the added noise (Proposition 5.4)

To allow us to apply the convergence analysis of Kairouz et al. [27] as a black box, we assume: (1) Θ is convex, and $\ell(\cdot; x)$ is convex for every choice of x; (2) ℓ is 1-Lipschitz in h (that is, suppose $\Theta \in \mathbb{R}^d$ and for all h, h' and x, we have $|\ell(h; x) - \ell(h'; x)| \leq \|h - h'\|_2$); (3) One new user arrives between adjacent ticks, though deletions may occur at any time.

Proposition 5.5 (Derived from Theorem 5.1 from *arXiv v3* of Kairouz et al. [27])**.** *There is an (ϵ, δ)-adaptively pan-private, continual-release algorithm that takes takes a time horizon T, a learning rate $\lambda > 0$, and a sequence of inputs $x_1, x_2, \ldots$ and at the t-th* tick *releases a model h_t such that, for every $t \in [T]$, data set $\vec{x}_t \in \mathcal{Z}^n$ (consisting of records received by tick t), $h^* \in \Theta$, and $\beta > 0$, with probability at least $1 - \beta$ over the coins of the algorithm,*

$$\mathcal{L}_{\vec{x}_t}(h_t) - \mathcal{L}_{\vec{x}_t}(h^*) = O\left(\frac{\lambda \|h^*\|^2}{n} + \frac{\sqrt{d \ln(1/\delta) \ln(T) \ln(n/\beta)}}{\epsilon\lambda}\right).$$

This result immediately yields a controller that satisfies (ϵ, δ)-deletion as control and maintains a model whose accuracy tracks that of the best model trained on all the arrivals so far (where repeated arrivals of the same user are ignored). The version above uses a fixed learning rate λ which can be tuned to get error $\tilde{O}(d^{1/4}/\sqrt{n})$ for any particular n; however, one could also decrease the learning rate as $1/\sqrt{t}$ to get bounds that hold for all data set sizes.

In contrast to the HI-style algorithms proposed for machine unlearning [23], this controller need not update the model when a user is deleted.

6 PARALLEL COMPOSITION

In this section, we show that our definition does more than unify approaches to deletion based on history independence, confidentiality, and differential privacy. We describe a controller that satisfies deletion-as-control but none of the other three notions. To prove that the controller satisfies deletion-as-control, we prove that the definition enjoys a limited form of *parallel composition*: a controller that is built from two constituent sub-controllers, each of which is run independently (Figure 4), will be deletion compliant if the component controllers are compliant and satisfy additional conditions. We focus on a special case that suffices for our needs, and leave a general treatment of composition of deletion-as-control for future work.

To anchor the discussion, we consider the touchstone composed controller "Public Directory with DP Statistics," described in Algorithm 1. It provides a public directory (e.g., a phone book) which, in addition to answering directory queries, periodically reports the total number of users that have made queries to the directory. We build the touchstone controller C from two SHI dictionaries $\mathcal{D}$ and $\mathcal{U}$ (Section 3.1.1) and the pan-private tree mechanism $\mathcal{M}$ (Proposition 5.4). $\mathcal{D}$ implements the directory functionalities of reading and writing. $\mathcal{M}$ keeps track of an approximate count of users—current and former—that have looked up an entry in $\mathcal{D}$ for

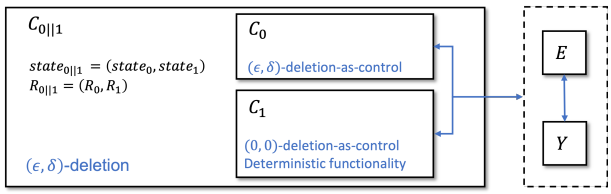

Figure 4: Parallel composition of controllers C_0 and C_1.

the first T epochs (e.g., the time between ticks). $\mathcal{U}$ is an auxiliary dictionary used to ensure that $\mathcal{M}$ counts distinct users.

Algorithm 1 Public directory with DP statistics

procedure INITIALIZE($T \in \mathbb{N}, \epsilon, \delta$)
  Initialize two SHI dictionaries $\mathcal{D}$ for the directory and $\mathcal{U}$ for users. Initialize the tree mechanism $\mathcal{M}$ with parameters (ϵ, δ, T)

procedure ACTIVATE(cID, op(arg))
  if op = delete **then**
    $\mathcal{D}$.delete(cID)
    $\mathcal{U}$.delete(cID)
    return $\perp$
  if op = set **then**
    $\mathcal{D}$.set(cID, arg)
    return $\perp$
  if op = get **then**
    if cID $\notin \mathcal{U}$ **then**
      $\mathcal{U}$.set(cID, 1)
      Insert the value 1 into $\mathcal{M}$'s data stream
    return $\mathcal{D}$.get(arg)
  if op = getCount **then**
    return $\mathcal{M}$(tick)

To prove that C satisfies deletion-as-control, we prove that one can build deletion-as-control controllers from two constituent sub-controllers by parallel composition. We consider the special case where one sub-controller satisfies $(0, 0)$-deletion-as-control and implements a *deterministic functionality*, and where both sub-controllers are *query-response controllers*. We define these next. Note that the controllers given in Section 3.2 and Section 5 are all query-response controllers. While we restrict our attention to a single controller of each type, the result immediately extends by induction to many query-response controllers satisfying $(0, 0)$ deletion-as-control with deterministic functionalities.

Definition 6.1 (Deterministic functionality)**.** *A controller* C *implements a* deterministic functionality *if for all* E, Y*, the transcript of* $\langle \mathsf{C}(R_\mathsf{C}), \mathsf{E}, \mathsf{Y} \rangle$ *is independent of* R_C.

Definition 6.2 (Query-response controller)**.** C *is a* query-response *controller if it always replies to the same party that activated it. Namely, when* C *is activated with* (cID, msg) *on its input tape, it always halts with* (cID, msg′) *on its output tape for some* msg′.

The example controllers we discuss are generally query-response. But a messaging server, for example, might not be: a request from Alice to send something Bob could result in a push from the server to Bob.

Definition 6.3 (Parallel composition)**.** *Let* C_0 *and* C_1 *be controllers. We define their* parallel composition*, denoted* $\mathsf{C}_{0\|1}$*, to be the controller that emulates* C_0 *an* C_1 *internally. When activated with* op_{id}*,* $\mathsf{C}_{0\|1}$ *computes* $\mathsf{out}_0 \leftarrow \mathsf{C}_0(\mathsf{op}_{id})$ *and* $\mathsf{out}_1 \leftarrow \mathsf{C}_1(\mathsf{op}_{id})$ *and returns* $\mathsf{out}_{0\|1} = (\mathsf{out}_0, \mathsf{out}_1)$*. The controllers* C_0 *and* C_1 *are emulated with disjoint portions of* $\mathsf{C}_{0\|1}$*'s randomness tape* $R_{0\|1} = (R_0, R_1)$ *and work tape* $\mathsf{state}_{0\|1} = (\mathsf{state}_0, \mathsf{state}_1)$.

THEOREM 6.4. *Let* C_0 *and* C_1 *be query-response controllers and let* $C_{0\|1}$ *be their parallel composition. If* C_0 *satisfies* (ϵ, δ)*-deletion-as-control, and if* C_1 *satisfies* $(0, 0)$*-deletion-as-control and has deterministic functionality, then* $\mathsf{C}_{0\|1}$ *satisfies* (ϵ, δ)*-deletion-as-control.*

Corollary 6.5. *The Public Directory with DP Statistics (defined formally in [10]) satisfies* (ϵ, δ)*-deletion-as-control.*

PROOF. Let C_0 be the controller $\mathsf{C}^{\mathsf{pp}}_{\mathcal{M}}$ defined relative to the pan-private tree mechanism $\mathcal{M}$. Let C_1 be the controller relative to a SHI dictionary (Definition 3.5). By construction, C_0 and C_1 are query-response controllers and C_1 has a deterministic functionality. C_0 satisfies (ϵ, δ) deletion-as-control and C_1 satisfies (ϵ, δ) deletion-as-control (Theorems 5.3 and 3.6 respectively). The Public Directory with DP Statistics is equivalent to the parallel composition $\mathsf{C}_{0\|1}$ of C_0 and C_1. By parallel composition, the Public directory with DP statistics satisfies (ϵ, δ) deletion-as-control. □

7 CONCLUSION

Defining deletion-as-control in a way that is both expressive and meaningful is the central challenge of our work. We believe that our definition succeeds, providing a new perspective to the ongoing discussion on how to give users control over their data.

More work is needed to understand how deletion-as-control handles the complexity of real-life functionalities. For example, we are far from understanding the implications for something like Twitter, though the Public Bulletin Board serves as a starting point. Analyzing complex functionalities may involve further studying the adaptive variants of history independence and pan-privacy defined in this work. Beyond specific functionalities, more work is needed to interpret the guarantees provided by our notion. In particular, we have a limited view of what can be said about groups of individuals and about composition.

These are directions for future technical work, but also for normative, legal, and policy considerations. Consider, for example, the goal of machine unlearning: maintaining a model while respecting requests to delete. The algorithms in the machine unlearning literature seek to approximate what one would get by retraining from scratch (that is, history independence). But using adaptive pan-privacy, one can satisfy deletion-as-control without updating the model in response to deletion requests. Each behavior might be appropriate for a different setting, depending on the most relevant measure of model accuracy. The difference may also relate to whether one adopts an individual- or group-based view of a right to erasure. We hope that our work inspires further exploration of these questions.

ACKNOWLEDGMENTS

We are grateful to helpful discussions with many colleagues, notably Kobbi Nissim. A.C. by the National Science Foundation under Grant No. 1915763 and by the DARPA SIEVE program under Agreement No. HR00112020021. A.S. and M.S. were supported by NSF awards CCF-1763786 and CNS-2120667, as well as faculty research awards from Google and Apple. P.V. was supported by the National Research Foundation, Singapore, under its NRF Fellowship programme, award no. NRF-NRFF14-2022-0010.

REFERENCES

[1] A. Achille, M. Kearns, C. Klingenberg, and S. Soatto. Ai model disgorgement: Methods and choices. *arXiv preprint arXiv:2304.03545*, 2023.

[2] M. Altman, A. Cohen, K. Nissim, and A. Wood. What a hybrid legal-technical analysis teaches us about privacy regulation: The case of singling out. *BUJ Sci. & Tech. L.*, 27:1, 2021.

[3] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

[4] G. E. Blelloch and D. Golovin. Strongly history-independent hashing with applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007. doi: 10.1109/FOCS.2007.36.

[5] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy*, 2021. doi: 10.1109/SP40001.2021.00019.

[6] Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

[7] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[8] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):1–24, 2011.

[9] A. Cohen and K. Nissim. Linear program reconstruction in practice. *Journal of Privacy and Confidentiality*, 10(1), Jan. 2020. doi: 10.29012/jpc.711. URL https://journalprivacyconfidentiality.org/index.php/jpc/article/view/711.

[10] A. Cohen, A. Smith, M. Swanberg, and P. N. Vasudevan. Control, confidentiality, and the right to be forgotten. *arXiv preprint arXiv:2210.07876*, 2022.

[11] Court of Justice of the European Union. Press release no 70/14. https://curia.europa.eu/jcms/upload/docs/application/pdf/2014-05/cp140070en.pdf, May 2013.

[12] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010*, pages 715–724. ACM, 2010. doi: 10.1145/1806689.1806787.

[13] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and S. Yekhanin. Pan-private streaming algorithms. In *ICS*, pages 66–80, 2010.

[14] B. Fowler. Data breaches break record in 2021. *CNET*, 2022. URL https://www.cnet.com/news/privacy/record-number-of-data-breaches-reported-in-2021-new-report-says/.

[15] S. Garg, S. Goldwasser, and P. N. Vasudevan. Formalizing data deletion in the context of the right to be forgotten. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*, pages 373–402. Springer, 2020.

[16] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019.

[17] J. Godin and P. Lamontagne. Deletion-compliance in the absence of privacy. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE, 2021.

[18] A. Golatkar, A. Achille, and S. Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020.

[19] A. Golatkar, A. Achille, and S. Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. *arXiv preprint arXiv:2003.02960*, 2020.

[20] Google. Right to be forgotten overview. https://support.google.com/legal/answer/10769224. Accessed: 2023-04-18.

[21] Google. Helping public health officials combat covid-19. https://blog.google/technology/health/covid-19-community-mobility-reports/, 2020. Accessed: 2023-04-18.

[22] C. Guo, T. Goldstein, A. Hannun, and L. van der Maaten. Certified data removal from machine learning models, 2020.

[23] V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites. Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34: 16319–16330, 2021.

[24] J. D. Hartline, E. S. Hong, A. E. Mohr, W. R. Pentney, and E. C. Rocke. Characterizing history independent data structures. *Algorithmica*, 42(1):57–74, 2005.

[25] P. Jain, S. Raskhodnikova, S. Sivakumar, and A. D. Smith. The price of differential privacy under continual observation. *CoRR*, abs/2112.00828, 2021. URL https://arxiv.org/abs/2112.00828.

[26] JASON. Consistency of data products and formal privacy methods for the 2020 Census. Panel Report JSR 21-02, JASON, The MITRE Corporation, 1 2022.

[27] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. Practical and private (deep) learning without sampling or shuffling. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[28] G. King and N. Persily. Unprecedented facebook urls dataset now available for academic research through social science one. https://socialscience.one/blog/unprecedented-facebook-urls-dataset-now-available-research-through-social-science-one, 2020. Accessed: 2023-04-18.

[29] D. Micciancio. Oblivious data structures: applications to cryptography. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 456–464, 1997.

[30] M. Naor and V. Teague. Anti-persistence: History independent data structures. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 492–501, 2001.

[31] National Conference of State Legislatures. State laws related to digital privacy. https://www.ncsl.org/technology-and-communication/state-laws-related-to-digital-privacy, June 2022. Accessed: 2023-04-18.

[32] S. Neel, A. Roth, and S. Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning, 2020.

[33] A. Ng. Homeland Security records show 'shocking' use of phone data, ACLU says. *Politico*, 2022. URL https://www.politico.com/news/2022/07/18/dhs-location-data-aclu-00046208.

[34] K. Nissim, A. Bembenek, A. Wood, M. Bun, M. Gaboardi, U. Gasser, D. R. O'Brien, T. Steinke, and S. Vadhan. Bridging the gap between computer science and legal approaches to privacy. *Harv. JL & Tech.*, 31:687, 2017.

[35] Y. Polyanskiy. Two fundamental probabilistic models. https://ocw.mit.edu/courses/6-436j-fundamentals-of-probability-fall-2018/resources/mit6_436jf18_lec02/, 2018. Accessed: 2023-04-18.

[36] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh. Remember what you want to forget: Algorithms for machine unlearning. *arXiv preprint arXiv:2103.03279*, 2021.

[37] R. K. Slaughter, J. Kopec, and M. Batal. Algorithms and economic justice: A taxonomy of harms and a path forward for the federal trade commission. *Yale JL & Tech.*, 23:1, 2020.

[38] A. Thudi, H. Jia, I. Shumailov, and N. Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4007–4022, 2022.

[39] E. Ullah, T. Mai, A. Rao, R. Rossi, and R. Arora. Machine unlearning via algorithmic stability. *arXiv preprint arXiv:2102.13179*, 2021.

[40] US Census Bureau. Why the Census Bureau chose differential privacy. Technical Report C2020BR-03, US Census Bureau, March 2023.

[41] S. Vadhan and W. Zhang. Concurrent composition theorems for all standard variants of differential privacy. *arXiv preprint arXiv:2207.08335*, 2022. To appear in the 55th Annual ACM Symposium on Theory of Computing (STOC), 2023.