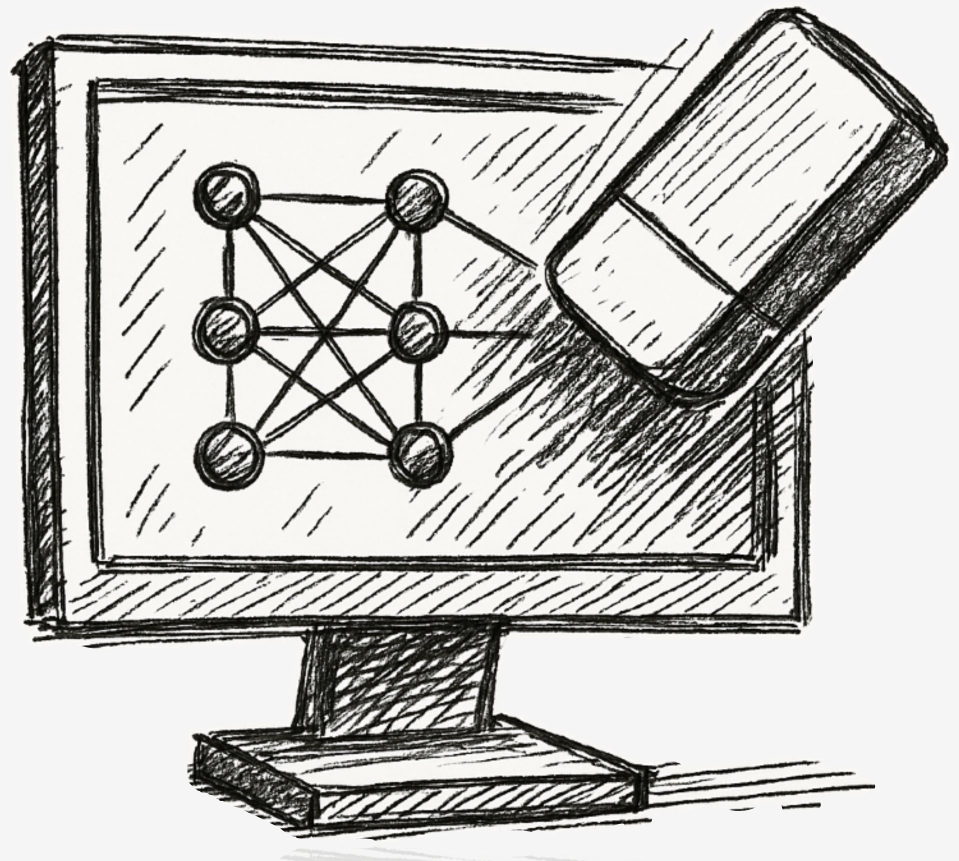


Verifiable and Provably Secure Machine Unlearning

Thorsten Eisenhofer [1], Doreen Riepel [2], Varun Chandrasekaran [3], Esha Ghosh [4], Olya Ohrimenko [5], and Nicolas Papernot [6]

[1] BIFOLD & TU Berlin, [2] CISPA Helmholtz Center for Information Security,
[3] University of Illinois Urbana-Champaign, [4] Microsoft Research,
[5] The University of Melbourne, [6] University of Toronto & Vector Institute

Machine Unlearning

Server 

Share point ■

$D = \{\blacksquare, \blacksquare, \blacksquare, \blacksquare, \dots\}$

$m \leftarrow \mathrm{Train}(D)$

Model might leak sth about its training data

Delete point ■

$D' = D \setminus \{\blacksquare\}$

$m' \leftarrow \mathrm{Unlearn}(m, \blacksquare)$

↯ Might not be done faithfully

ok!

Can we trust the server?

Goal: Prove that the unlearning actually happened

Proof via Model Parameters

Shumailov et al. "**Manipulating SGD with Data Ordering Attacks**", NeurIPS'21
Thudi et al. "**On the Necessity of Auditable Algorithmic Definitions for Machine Unlearning**", USENIX'20

Plausibility forgery

- ↯ **Possible to efficiently construct D' s.t. $m = m'$ but $D \neq D'$**
- ↯ **Always possible if D and D' share the underlying distribution**

Server

Share point ■ →

$D = \{■, ■, ■, ■, \dots\}$

$m \leftarrow \text{Train}(D)$

Delete point ■ →

$D' = D \setminus \{■\}$

$m' \leftarrow \text{Unlearn}(m, ■)$

← D', m'

If ■ $\in D'$: abort

$m \leftarrow \text{Train}(D)$

If $m \neq m'$: abort

One-Shot Verifiable Unlearning

Verifiable Computation

y := f(x)

Proof that **y** is the result of evaluating **f** on **x**

Server 

Share point ■

$D = \{■, ■, ■, ■, \ldots\}$

$m \leftarrow \text{Train}(D)$

Delete point ■

$D' = D \setminus \{■\}$

$m', \rho \leftarrow \text{ProveUnlearn}(m, ■)$

D', m', m, ρ

↯ **May be forged**

If ■ $\in D'$**: abort**

If not $\text{VerifyUnlearn}(m, m', ■, \rho)$**: abort**

How can we prove unlearning?

A Naive Iteration-based Protocol

Server 

Share point ■

$D = \{\blacksquare, \blacksquare, \blacksquare, \blacksquare, \dots\}$

$m, \tau \leftarrow \mathrm{ProveTrain}(D)$

D, m, τ

If not $\mathrm{VerifyTrain}(m, D, \tau)$ **: abort**

Delete point ■

$D' = D \setminus \{\blacksquare\}$

$m', \rho \leftarrow \mathrm{ProveUnlearn}(m, \blacksquare)$

D', m', ρ

Verify consistency of the dataset

If $D' \neq D \cup \{\blacksquare\}$**: abort**

If not $\mathrm{VerifyUnlearn}(m, m', \blacksquare, \rho)$ **: abort**

A Naive Iteration-based Protocol

Users $\mathcal{U}$ $\{\mathsf{pub},\ \widehat{D}_u \sim \mathcal{D}\}_{u\in\mathcal{U}}$		Server S (pub)
Initialize		
if not $\mathsf{VerifyInit}(\mathsf{pub}, \mathsf{com}_0, \rho_0)$:	$\xleftarrow{\mathsf{com}_0,\ \rho_0}$	$(\mathsf{st}_{S,0}, m_0, \mathsf{com}_0, \rho_0) \leftarrow \mathsf{Init}(\mathsf{pub})$
abort		$D_0^+ := \emptyset,\ U_0^+ := \emptyset$
i-th iteration		$D_i^+ := D_{i-1}^+,\ U_i^+ := U_{i-1}^+$
# add data points k-th query	$\xrightarrow{u\in\mathcal{U},\ d_{i,k}\in\widehat{D}_u}$	$D_i^+ := D_i^+ \cup \{(u, d_{i,k})\}$
# remove data points j-th query	$\xrightarrow{u\in\mathcal{U},\ d_{i,j}\in\widehat{D}_u}$	$U_i^+ := U_i^+ \cup \{(u, d_{i,j})\}$
Proof of Training		
if not $\mathsf{VerifyTraining}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$	$\xleftarrow{\textit{train:}\ \mathsf{com}_i,\ \rho_i}$	$(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveTraining}(\mathsf{st}_{S,i-1}, \mathsf{pub}, D_i^+)$
abort		$D_i^+ := \emptyset$
OR Proof of Unlearning		
if not $\mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$:	$\xleftarrow{\textit{unlearn:}\ \mathsf{com}_i,\ \rho_i}$	$(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveUnlearning}(\mathsf{st}_{S,i-1}, \mathsf{pub}, U_i^+)$
abort		
		for $(u, d_{i,j}) \in U_i^+$:
if not $\mathsf{VerifyNonMembership}(\mathsf{pub}, u, d_{i,j}, \mathsf{com}_i, \pi_{u,d_{i,j}})$:	$\xleftarrow{\pi_{u,d_{i,j}}}$	$\pi_{u,d_{i,j}} \leftarrow \mathsf{ProveNonMembership}(\mathsf{st}_{S,i}, \mathsf{pub}, u, d_{i,j})$
abort		$U_i^+ := \emptyset$

Multiple users

Multiple iterations

Share only *commitment* on model and data

If not V

Verify consi
of the dat

If $D' \neq D \cup \{\blacksquare\}$: abort

If not VerifyUnlearn$(m, m', \blacksquare, \rho)$: ab

Framework for Verifiable Unlearning

Security Definition

$\mathsf{GameUnlearn}_{\mathcal{A},\mathcal{E},\Phi_f,\mathcal{D}}(1^\lambda)$

$\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$

$(k, (u,d), \pi_{u,d}, \{mode_i\colon \mathsf{com}_i, \rho_i\}_{i\in[0:\ell]};$
$\{D_i\}_{i\in[0:\ell]}) \leftarrow (\mathcal{A}\|\mathcal{E})(\mathsf{pub}, \mathsf{aux})$

Run adversary $\mathscr{A}$

 # Pre-proces...

$U_k^+ := D_{k-1} \setminus D_k$

Parse com_i as $(\mathsf{com}_i^m \| \mathsf{com}_i^D)$ $\forall i \in [0:\ell]$

 # Evaluate winning condition

if $\mathsf{Commit}(\mathsf{pub}, D_i) = \mathsf{com}_i^D$ $\forall i \in [0:\ell]$ *# Datasets*

and $\mathsf{VerifyInit}(\mathsf{pub}, \mathsf{com}_0, \rho_0)$ *# Initialization*

and $\mathsf{VerifyTraining}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$ $\forall i : mode_i = train$ *# Training*

and $\mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$ *# Unlearning*

Evaluate output of $\mathscr{A}$

and VerifyNonMembership(pub, u, ..., $\mathsf{com}_k, \pi_{u,d}$) *# Non-Membership*

and $(u,d) \in U_k^+$ *# Point unlearnt*

and $(u,d) \in D_\ell$ **and** $k < \ell$: *# Point re-added later*

return 1

return 0

Adversary wins if they can forge an unlearning response

Definition (Unlearning)

Let λ be the security parameter and consider game Gam... is unlear... secure if for all PPT adversaries $\mathscr{A}$ there exists an extractor $\mathscr{E}$ such that for all benign auxiliary inputs aux

A protocol is *unlearning-secure* if no efficient adversary exists that can forge an unlearning response.

$$\Pr[\mathsf{GameUnlearn}_{\mathscr{A},\mathscr{E},\Phi_f,\mathscr{D}}(1^\lambda) \Rightarrow 1] \le \mathsf{negl}(\lambda)$$

Prove completeness and security of protocols under this definition!

Take Aways

Verifiable unlearning

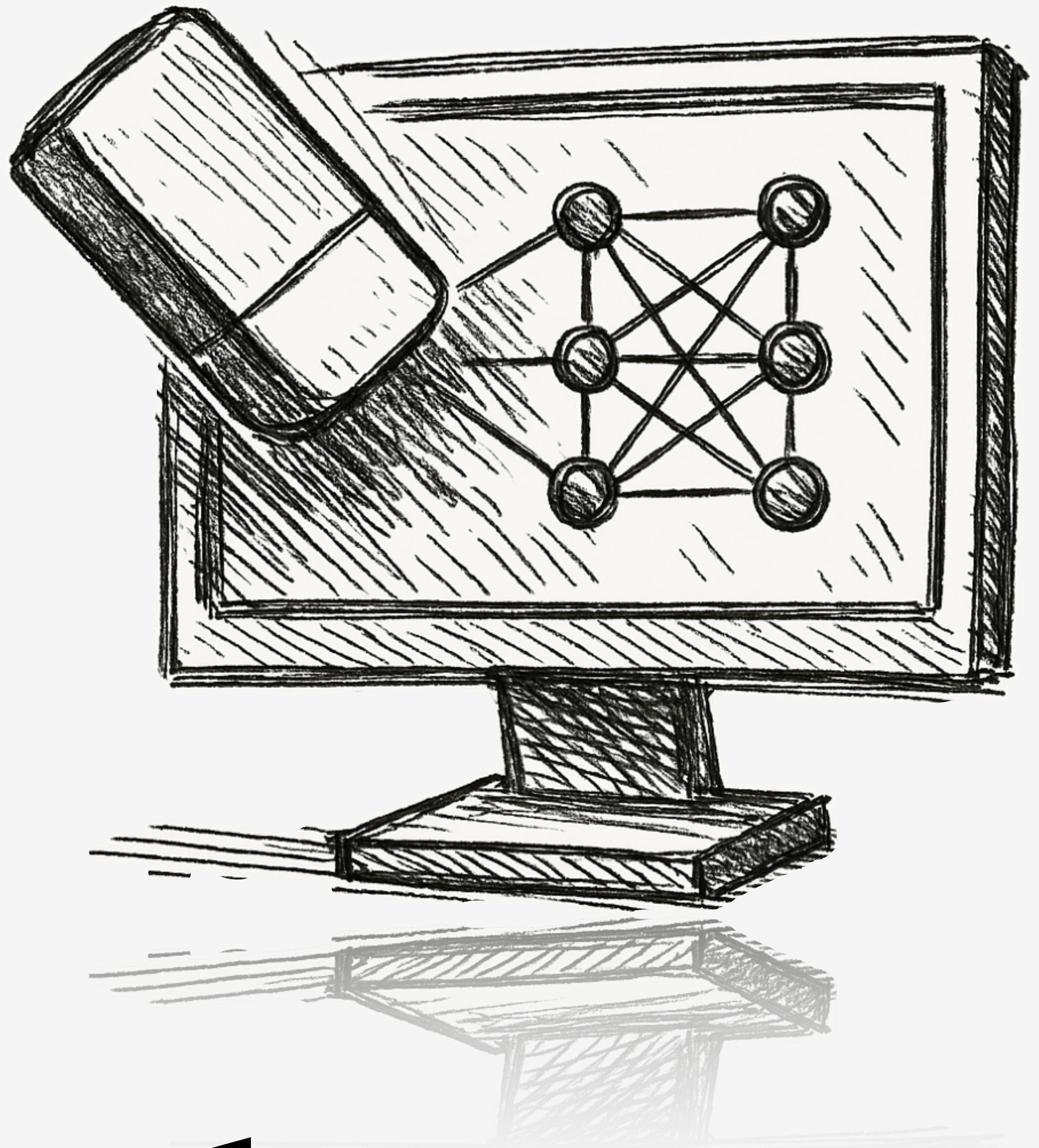
- Verify that unlearning algorithm was executed
- Consider lifecycle of the model

Security definition

- Instantiate protocol under this definition
- Allows to prove completeness and security

Paper and code

github.com/cleverhans-lab/verifiable-unlearning



Thank you!