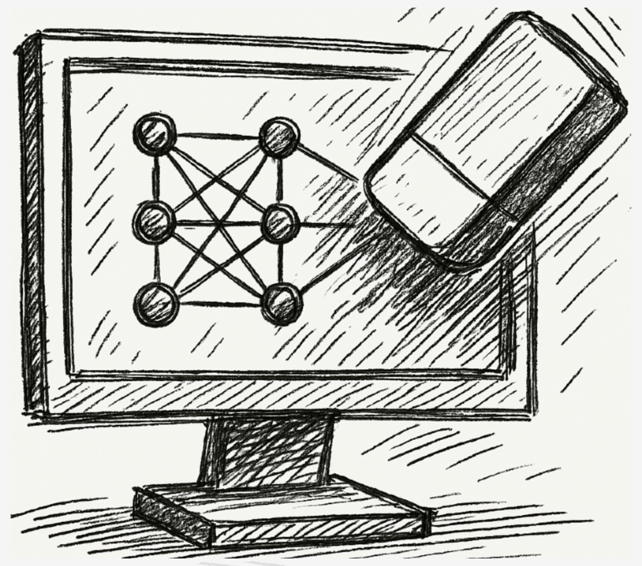

Verifiable and Provably Secure Machine Unlearning

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

Motivation

Server 

Shares ■ →

$D = \{■, ■, ■, ■, \dots\}$

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

Delete ■ →

$D' = D' \setminus \{■\}$

$m' \leftarrow \mathsf{Unlearn}(m, ■)$

← **ok!**

But can we trust the server?

Goal: Prove that unlearning was performed correctly!

Unlearning Framework

Proof via model parameters not sufficient. Can efficiently construct dataset D' s.t. $D' \neq D$ but $m = m'$ ↯

Verifiable Unlearning

Proof of unlearning
Verify correct execution of unlearning algorithm

Proof of training
Consider full lifecycle of the model

Users $\mathcal{U}$ $\{\mathsf{pub}, \widehat{D}_u \sim \mathcal{D}\}_{u \in \mathcal{U}}$		**Server** S (pub)
Initialize		
if not $\mathsf{VerifyInit}(\mathsf{pub}, \mathsf{com}_0, \rho_0)$: **abort**	← com_0, ρ_0	$(\mathsf{st}_{S,0}, m_0, \mathsf{com}_0, \rho_0) \leftarrow \mathsf{Init}(\mathsf{pub})$; $D_0^+ := \emptyset,\ U_0^+ := \emptyset$
i-th iteration		$D_i^+ := D_{i-1}^+,\ U_i^+ := U_{i-1}^+$
# add data points k-th query	$u \in \mathcal{U},\ d_{i,k} \in \widehat{D}_u$ →	$D_i^+ := D_i^+ \cup \{(u, d_{i,k})\}$
# remove data points j-th query	$u \in \mathcal{U},\ d_{i,j} \in \widehat{D}_u$ →	$U_i^+ := U_i^+ \cup \{(u, d_{i,j})\}$
Proof of Training		
if not $\mathsf{VerifyTraining}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$ **abort**	← *train:* com_i, ρ_i	$(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveTraining}(\mathsf{st}_{S,i-1}, \mathsf{pub}, D_i^+)$; $D_i^+ := \emptyset$
OR Proof of Unlearning		
if not $\mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$: **abort**	← *unlearn:* com_i, ρ_i	$(\mathsf{st}_{S,i}, m_i, \mathsf{com}_i, \rho_i) \leftarrow \mathsf{ProveUnlearning}(\mathsf{st}_{S,i-1}, \mathsf{pub}, U_i^+)$
if not $\mathsf{VerifyNonMembership}(\mathsf{pub}, u, d_{i,j}, \mathsf{com}_i, \pi_{u,d_{i,j}})$: **abort**	← $\pi_{u,d_{i,j}}$	**for** $(u, d_{i,j}) \in U_i^+$: $\pi_{u,d_{i,j}} \leftarrow \mathsf{ProveNonMembership}(\mathsf{st}_{S,i}, \mathsf{pub}, u, d_{i,j})$; $U_i^+ := \emptyset$

Security Definition

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

$\mathsf{pub} \leftarrow \mathsf{Setup}(1^\lambda)$
$(k, (u,d), \pi_{u,d}, \{mode_i\colon \mathsf{com}_i, \rho_i\}_{i \in [0:\ell]};\ \{D_i\}_{i \in [0:\ell]}) \leftarrow (\mathcal{A} \| \mathcal{E})(\mathsf{pub}, \mathsf{aux})$
 # Pre-processing
$U_k^+ := D_{k-1} \setminus D_k$
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 $\mathsf{VerifyUnlearning}(\mathsf{pub}, \mathsf{com}_{i-1}, \mathsf{com}_i, \rho_i)$ $\forall i : mode_i = unlearn$ *# Unlearning*
  and $\mathsf{VerifyNonMembership}(\mathsf{pub}, u, d, \mathsf{com}_k, \pi_{u,d})$ *# 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)

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

Instantiation

Prove correct execution of training and unlearning algorithm using techniques from verifiable computation.

Verifiable computation

$\mathbf{y := f(x)}$ *"Proof that y is the result of evaluating f on x"*

C_U(**public** $h_{\mathsf{st}_{f,i}}, h_{\mathsf{st}_{f,i-1}}, h_{m_i}, h_{D_i}, h_{D_{i-1}}, h_{U_i}, h_{U_{i-1}}$,
private $\mathsf{st}_{f,i-1}, \mathcal{H}_{D_{i-1}}, U_i^+$)
 # Check input set of hashed training data records
if $h_{D_{i-1}} \neq \mathsf{HashData}(\mathcal{H}_{D_{i-1}})$:
  return false
 # Update and check set of hashed unlearnt data records and training data records
$\mathcal{H}_{U_i^+} := \{\mathsf{HashDataRecord}(u,d)\}_{(u,d) \in U_i^+}$
$\mathcal{H}_{D_i} := \mathcal{H}_{D_{i-1}} \setminus \mathcal{H}_{U_i^+}$
if $h_{U_i} \neq \mathsf{AppendHashData}(h_{U_{i-1}}, \mathcal{H}_{U_i^+})$ **or** $h_{D_i} \neq \mathsf{HashData}(\mathcal{H}_{D_i})$:
  return false
 # Check input state, perform unlearning and check outputs
$h_{\mathsf{st}_{f,i-1}} \neq \mathsf{HashState}(\mathsf{st}_{f,i-1})$:
  return false
$(\mathsf{st}_{f,i}, m_i) := f_U(\mathsf{st}_{f,i-1}, U_i^+)$
if $h_{\mathsf{st}_{f,i}} \neq \mathsf{HashState}(\mathsf{st}_{f,i})$ **or** $h_{m_i} \neq \mathsf{HashModel}(m_i)$:
  return false
return true