# Practical Secure Aggregation for Privacy-Preserving Machine Learning

# (Ch4∼)

이종진

Seoul National University

*ga0408@snu.ac.kr*

August 25, 2020

# Previous

- Multiparty computation, Federated learning setting.
- Considering training neural network to predict the next word.
  - Mobile devices, central server.
  - Communication is extremely expensive and user dropouts are common.
  - How to securely aggregate the data from mobile devices.

# Crptographic Primitives

- Secret Sharing(Shamir's $t$-out-of-$n$)
- Key Agreement($g, g^a, g^b, g^{ab}$)
- Authenticated Encryption
- Pseudorandom Generator(seed, $[0, R)^m$)
- Signature Scheme (for active adversary.)
- Public Key Infrastructure.

# How to aggregate the inputs

- Notation
  - A single server $\mathcal{S}$
  - $n$ client parties $\mathcal{U}$
  - A private vector for user $u \in \mathcal{U}$, $\boldsymbol{x_u}$, dimension $m$, in $\mathbb{Z}_R$ ($u \in \{1, \ldots, n\}$)
- The protocol can guarantee that the server only learns a sum of the clients' inputs.

# Masking with One-Time Pads

- Mask $\boldsymbol{x}_u$ in a particular way.
- Suppose each pair of users $(u, v)$, $u < v$ agree on some random vector $\boldsymbol{r}_{u,v}$

$$\boldsymbol{y}_u = \boldsymbol{x}_u + \sum_{v \in \mathcal{U}: u < v} \boldsymbol{r}_{u,v} - \sum_{v \in \mathcal{U}: u > v} \boldsymbol{r}_{v,u} \pmod{R}$$

- then,

$$\boldsymbol{z} = \sum \boldsymbol{y}_u = \sum \boldsymbol{x}_u \pmod{R}$$

# Masking with One-Time Pads

- Two shortcomings.
- The users must exchange the random vectors $\boldsymbol{r}_{u,v}$
  - (Requireq quadratic communication overhead($|\mathcal{U}| \times |\boldsymbol{x}|$))
- No tolerance for a party failing to complete the protocol.

# Efficient Communication and Handling Dropped Users.

- Reduce the communication by handling the parties agree on common seed($\boldsymbol{s}_{u,v}$) for PRG rather than on the entire mask $\boldsymbol{r}_{u,v}$
  - Seed may have comparatively low dimension
- Notify the surviving users of the drop-out, and to have them each reply with the common seed they computed with the dropped users.
  - Additional users may drop out in the recovery phase before replying with the seed.
  - Leading the number of rounds up to at most the number of users.
  - They use a threshold secret sharing scheme.

# Double-Masking to Protect Security

- Each user $u$ distributes shares of $\boldsymbol{s}_{u,v}$ to each of the other users.
- Secret sharing scheme allows dropped user's seed to be recovered
- This approach can solve the problem of unbounded recovery rounds, but still has issue.
  - If user $u$'s device is too slow in sending her $\boldsymbol{y}_u$ to the server.
  - Adversarial server in the active model can learn $\boldsymbol{x}_u$ by lying about whether user $u$ has dropped out.
- Double masking.

# Double-Masking to Protect Security

- To resolve this new security problem, they introduce doubly masking structure.
- Each user $u$ samples an additional random seed $\boldsymbol{b}_u$ and distributes shares of $\boldsymbol{b}_u$ to each of the other users.

$$\boldsymbol{y}_u = \boldsymbol{x}_u + \boldsymbol{PRG}(\boldsymbol{b}_u) + \sum_{v \in \mathcal{U}: u < v} \boldsymbol{PRG}(\boldsymbol{s}_{u,v}) - \sum_{v \in \mathcal{U}: u > v} \boldsymbol{PRG}(\boldsymbol{s}_{v,u}) \pmod{R}$$

- Dropped user's $\boldsymbol{s}_{u,v}$ and surviving user's $\boldsymbol{b}_u$ are needed.

# Secure Aggregation Protocol

- They present a protocol which has a constant number of rounds, low communication overhead, robustness to failures.
- The protocol consists of 4 rounds.
- They present two variants of the protocol
  - Secure against honest but curious adversaries.
  - Secure against active adversaries.

# Secure Aggregation Protocol, Setup

- **Setup**
  - The protocol is run between a server($\mathcal{S}$) and a set of n users.
  - $\boldsymbol{x}_u \in \mathbb{Z}_R^m$ is a input vector for user $u$
  - The security parameter $k$, a threshold value $t$, and honestly generated $pp \leftarrow \boldsymbol{KA.gen(k)}$
  - The server can communicate with the users with secure channels. (public keys, encoded secret keys)
  - $\underline{\text{All users } u \text{ receive their signing key } d_u^{SK} \text{ from the trusted third party,}}$ $\underline{\text{together with verification keys } d_v^{PK} \text{ bound to each user identity } v}$

# Secure Aggregation Protocol, Round 0 (AdvetiseKeys)

- **User u:**
  - Generate key pairs $\left(c_u^{PK}, c_u^{SK}\right) \leftarrow \boldsymbol{KA.gen}(pp)$, $\left(s_u^{PK}, s_u^{SK}\right) \leftarrow \boldsymbol{KA.gen}(pp)$, $\underline{\sigma_u \leftarrow \boldsymbol{SIG.sign}\left(d_u^{SK}, c_u^{PK} \| s_u^{PK}\right)}$
  - Send $\left(c_u^{PK} || s_u^{PK} || \underline{\sigma_u}\right)$ to the server.
- **Server:**
  - Denote users set $\mathcal{U}_1$ in this round
  - (Assert that $|\mathcal{U}_1| \geq t$)
  - Broadcast to all users in $\mathcal{U}_1$ the list $\{(v, c_v^{PK}, s_v^{PK}, \underline{\sigma_v})\}$

# Secure Aggregation Protocol, Round 1 (ShareKeys)

- **User u:**
  - Received the list $\{(v, c_v^{PK}, s_v^{PK}, \underline{\sigma_v})\}$ $\underline{\text{Verify that } \forall v \in \mathcal{U}_1,\ \text{SIG.ver}\ \left(d_v^{PK}, c_v^{PK} \| s_v^{PK}, \sigma_v\right) = 1}$
  - Sample a random element $\boldsymbol{b}_u \leftarrow \mathbb{F}$
  - Generate $t$-out-of $|\mathcal{U}_1|$ shares of $\boldsymbol{s}_u^{SK}$ and $\boldsymbol{b}_u$: $\left\{\left(v, \boldsymbol{s}_{u,v}^{SK}\right)\right\}_{v \in \mathcal{U}_1} \leftarrow \boldsymbol{SS.share}\left(\boldsymbol{s}_u^{SK}, t, \mathcal{U}_1\right),$ $\{(v, b_{u,v})\}_{v \in \mathcal{U}_1} \leftarrow \boldsymbol{SS.share}\left(\boldsymbol{b}_u, t, \mathcal{U}_1\right)$
  - Encode secret keys for each other user $v \in \mathcal{U}_1 \setminus \{u\}$ ,Compute $e_{u,v} \leftarrow \boldsymbol{AE.enc}(\boldsymbol{KA.agree}(c_u^{SK}, c_v^{PK}), u||v||s_{u,v}^{SK}||b_{u,v})$
  - Send all the cipertexts $\{e_{u,v}\}_{v \in \mathcal{U}_2}$ to the server
- **Server:**
  - Collect lists of ciphertexts from at least $t$ users (denote $\mathcal{U}_2 \subset \mathcal{U}_1$)
  - (Assert that $|\mathcal{U}_2| \geq t$)
  - Sent to each user $u \in U_2$ all ciphertexts $\{e_{u,v}\}_{v \in \mathcal{U}_2}$

# Secure Aggregation Protocol, Round 2 (MaskedInputCollection)

- **User u:**
  - For each other user $v \in \mathcal{U}_2 \setminus \{u\}$ compute $\boldsymbol{s}_{u,v} \leftarrow \boldsymbol{KA.agree}(s_u^{SK}, s_v^{PK})$
  - Compute $\boldsymbol{p}_{u,v}$

$$\boldsymbol{p}_{u,v} = \begin{cases} \boldsymbol{PRG}\left(s_{u,v}\right), & \text{when, } u > v \\ \boldsymbol{PRG}\left(s_{u,v}\right), & \text{when, } u < v \\ \boldsymbol{PRG}\left(s_{u,v}\right), & \text{when, } u = v \end{cases}$$

  - Compute $\boldsymbol{p}_u = \boldsymbol{PRG}(b_u)$
  - Compute $\boldsymbol{y}_u = \boldsymbol{x}_u + \boldsymbol{p}_u + \sum_{v \in \mathcal{U}_2} \boldsymbol{p}_{u,v}$
- **Server:**
  - Collect $\boldsymbol{y}_u$ from at least t users. (denote $\mathcal{U}_3 \subset \mathcal{U}_2$)

# Secure Aggregation Protocol, Round 3 (ConsistencyCheck)

- **User u:**
  - $\underline{\text{Send to the server } \sigma_u' \leftarrow \boldsymbol{SIG.sign}(d_u^{SK}, \mathcal{U}_3)}$
- **Server:**
  - $\underline{\text{ Collect } \sigma_u' \text{ from at least } t \text{ users (denote } \mathcal{U}_4 \subset \mathcal{U}_3\text{) }}$, $\underline{\text{ Send to each user in } \mathcal{U}_4 \text{ the set } \{v, \sigma_v'\}_{v \in \mathcal{U}_\mathbf{4}}}$

# Secure Aggregation Protocol, Round 4 (Unmasking)

- **User u:**
  - $\underline{\boldsymbol{SIG.ver}(d^{PK}, \mathcal{U}_3, \sigma_v^{'}) = 1 \text{ for all } v \in \mathcal{U}_4}$
  - Assert that $u = u^{'}$ and $v = v^{'}$
  - Descrypt the ciphertext $v^{'} || u^{'} || s_{u,v}^{SK} || b_{v,u} \leftarrow \boldsymbol{AE.dec}(\boldsymbol{KA.agree}(c_u^{PK}, c_v^{PK}), e_{v,u})$
  - Send a list of shares to the server, which consists of $\boldsymbol{s}_{u,v}^{SK}$ for users $v \in \mathcal{U}_2 \setminus \mathcal{U}_3$ and $b_{v,u}$ for users in $v \in \mathcal{U}_3$
- **Server:**
  - Collect responses from at least $t$ users (denote with $\mathcal{U}_5$ this set of users).
  - For $u \in \mathcal{U}_2 \setminus \mathcal{U}_3$, reconstruct $s_{v,u} \leftarrow \boldsymbol{SS.recon}(\{s_{u,v}^{SK}\}_{v \in \mathcal{U}_\mathbf{5}}, t)$ for all $v \in \mathcal{U}_3$.
  - Compute $\boldsymbol{p}_{v,u}$ for all $v \in \mathcal{U}_2 \setminus \mathcal{U}_3$.
  - For $u \in \mathcal{U}_3$, reconstruct $b_u \leftarrow \boldsymbol{SS.recon}(\{b_{u,v\, v \in \mathcal{U}_\mathbf{5}}, t)\}$ and then recompute $\boldsymbol{p}_u$ using the PRG.
  - Compute and output $\boldsymbol{z} = \sum_{u \in \mathcal{U}_\mathbf{3}} \boldsymbol{x}_u$ as $\sum_{u \in \mathcal{U}_\mathbf{3}} \boldsymbol{x}_u = \sum_{u \in \mathcal{U}_\mathbf{3}} \boldsymbol{y}_u - \sum_{u \in \mathcal{U}_\mathbf{3}} \boldsymbol{p}_u + \sum_{u \in \mathcal{U}_\mathbf{3}, v \in \mathcal{U}_\mathbf{2} \setminus \mathcal{U}_\mathbf{3}} \boldsymbol{p}_{v,u}$

The end.