(In)security of CKKS-based Approximate Protocols

Intak Hwang[1], Yisol Hwang[1], Miran Kim[2], Dongwon Lee[1], Yongsoo Song[1]
[1] Seoul National University [2] Hanyang niversity

Summary

✓ **Insecurity of previous CKKS-based Protocols**

Existing CKKS-based protocols in which each party samples independent smudging noise fail to satisfy standard simulation-based MPC security notion.

✓ **Secure Protocols with Collaborative Sampling**

Our new collaborative sampling shifts the noise generation from a local process to a joint manner to solve this issue.

✓ **Approximate MPC with Relaxed Security**

We generalize the notion of liberal security and prove existing CKKS-based protocols satisfy this relaxed security.

Simulation-based MPC Security

✓ **Definition**

Π: n-party protocol computing a functionality $f = (f_1, \dots, f_n)$

$I \subseteq \{1, \dots, n\}$: set of parties' indices / $\vec{x} = (x_1, \dots, x_n)$: input

$view_i^{\Pi}$: tuple of P_i's input, randomness, and received messages

$output_i^{\Pi}$: P_i's output of the protocol

There exists PPT algorithms Sim^{Π} such that for every A and $\vec{x}$,

$$\left\{\left(Sim^{\Pi}\left(A, (x_i)_{i\in A}, f_A(\vec{x})\right), f(\vec{x})\right)\right\} \approx_c \left\{\left(view_A^{\Pi}(\vec{x}), output^{\Pi}(\vec{x})\right)\right\}$$

Security Analysis of Existing Protocols

✓ **Existing Protocols (2-Party)**

Client (P_1) and server (P_2) sample and add flooding noise e_1, e_2 to ensure $INDCPA^D$ security and circuit privacy, respectively.

Protocol Π_{Prev2PC}

(1) P_1 generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and encrypts $\mathsf{ct}_{\mathsf{in}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$. P_1 sends $(\mathsf{pk}, \mathsf{ct}_{\mathsf{in}})$ to P_2.
(2) P_2 computes $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{Eval}_{\mathsf{evk}}(C, \mathsf{ct}_{\mathsf{in}}, y)$, $\mathsf{ct}_{\mathsf{zero}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)$, and samples $e_2 \leftarrow \mathcal{D}$. P_2 computes $\mathsf{ct}'_{\mathsf{out}} := \mathsf{ct}_{\mathsf{out}} + \mathsf{ct}_{\mathsf{zero}} + (e_2, 0)$, and sends it to P_1.
(3) P_1 samples $e_1 \leftarrow \mathcal{D}$ and computes $z_{\mathsf{out}} := \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}'_{\mathsf{out}}) + e_1$, P_1 sends z_{out} to P_2, and both parties output z_{out}.

$view_1^{\Pi} = (\dots, ct'_{out}, e_1)$, $output_1^{\Pi} = z_{dec} + e_1 + e_2 (= z_{out})$

→ Client's view (e_1) influences the final output ($z_{dec} + e_1 + e_2$), a simulator cannot generate a client view consistent with using only the input and output.

→ If it could, it would effectively recover $z_{dec} + e_2$, revealing more information about exact result than allowed by the ideal functionality, which is a contradiction.

Secure Protocols with Collaborative Sampling

✓ **Collaborative Sampling**

Client (P_1) and server (P_2) sample and add flooding noise e_1, e_2 to ensure $INDCPA^D$ security and circuit privacy, respectively.

Functionality $\mathcal{F}_{\text{ColSamp}}$

Setup:
Let $\mathcal{D}$ be a smudging error distribution over R and q_{out} denote the modulus of output ciphertext.

1. Sample $e \leftarrow \mathcal{D}$.
2. Sample $r_1 \xleftarrow{\$} R_{q_{\mathsf{out}}}$ and compute $r_2 := -r_1 + e \pmod{q_{\mathsf{out}}}$.
3. Output r_i to P_i for $i = 1, 2$.

✓ **Secure CKKS-based Protocol**

Protocol Π_{App2PC}

Offline:
P_1 and P_2 call $\mathcal{F}_{\text{2-ColSamp}}$, and each party P_i obtains r_i for $i = 1, 2$.
.....
Online:

1. P_1 generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\mathsf{ct}_{\mathsf{in}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$. P_1 sends $(\mathsf{pk}, \mathsf{ct}_{\mathsf{in}})$ to P_2.
2. P_2 computes $\mathsf{ct}_{\mathsf{out}} \leftarrow \mathsf{Eval}_{\mathsf{evk}}(C, \mathsf{ct}_{\mathsf{in}}, y)$ and $\mathsf{ct}_{\mathsf{zero}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)$. P_2 computes $\mathsf{ct}_{\mathsf{rand}} := \mathsf{ct}_{\mathsf{out}} + \mathsf{ct}_{\mathsf{zero}} + (r_2, 0)$ and sends it to P_1.
3. P_1 computes $\hat{z} := \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}_{\mathsf{rand}} + (r_1, 0))$.
4. P_1 sends $\hat{z}$ to P_2. Both P_1 and P_2 output $\hat{z}$.

✓ **Security Proof**

$view_1^{\Pi} = (\dots, ct'_{out}, r_1)$, $output_1^{\Pi} = z_{dec} + e$ where $e = \sum r_i$

Since each r_i seems uniformly sample from $R_{q_{out}}$, both parties learn only the final output $z_{dec} + e$ and obtain no additional information about the exact functionality value.

Liberal Security for Approximate Computation

✓ **Liberal Security**

Π: n-party protocol computing a functionality $f = (f_1, \dots, f_n)$

$I \subseteq \{1, \dots, n\}$: set of parties' indices / $\vec{x} = (x_1, \dots, x_n)$: input

$view_i^{\Pi}$: tuple of P_i's input, randomness, received messages

$output_i^{\Pi}$: P_i's output of the protocol

$\hat{f} = (\hat{f}_1, \dots, \hat{f}_n)$: auxiliary information of functionality f

There exists Sim^{Π} such that for every A, $\mathrm{H} = [n] \backslash \mathrm{A}$, and $\vec{x}$,

$$\left\{\left(Sim^{\Pi}\left(A, (x_i)_{i\in A}, f_A(\vec{x}), \hat{f}_A(\vec{x})\right), f_H(\vec{x})\right)\right\} \approx_c \left\{\left(view_A^{\Pi}(\vec{x}), output_H^{\Pi}(\vec{x})\right)\right\}$$

✓ **Security Analysis of Previous Protocols**

✓ Simulator for P_1

(1) Generate $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\mathsf{ct}_{\mathsf{in}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$.
(2) Let $e := z - \hat{z}$ and sample $(e_1, e_2) \leftarrow \{(e_1, e_2) \leftarrow \mathcal{D} \times \mathcal{D} \mid e_1 + e_2 = e\}$.
(3) Sample $a_{\mathsf{out}} \xleftarrow{\$} R_{q_{\mathsf{out}}}$ and let $\mathsf{ct}'_{\mathsf{out}} := (b_{\mathsf{out}} := a_{\mathsf{out}} s + \hat{z} + e_2, a_{\mathsf{out}})$.
(4) Output $(x, \mathsf{sk}, \mathsf{pk}, \mathsf{ct}_{\mathsf{in}}, \mathsf{ct}'_{\mathsf{out}}, e_1)$.

When the auxiliary information is exact computation result, the simulator can extract $e = e_1 + e_2$ from the functionality output and auxiliary information. The view of adversarial party P_1 also can be simulated from this additional information.