TFMD

General and Fast Secure Neural Network Inference Framework with Threshold FHE

Yu Fu, Yu Tong, Yijing Ning, Jingqiang Lin, Dengguo Feng

IEEE Transactions on Information Forensics and Security · 2026

배경 지식: MPC (Secure Multi-Party Computation)

핵심 아이디어

각 파티가 자신의 데이터(x_i)를 공개하지 않고
협력하여 공동 함수 $f(x_1,\ldots,x_n)$를 계산

① Secret Sharing으로 분할

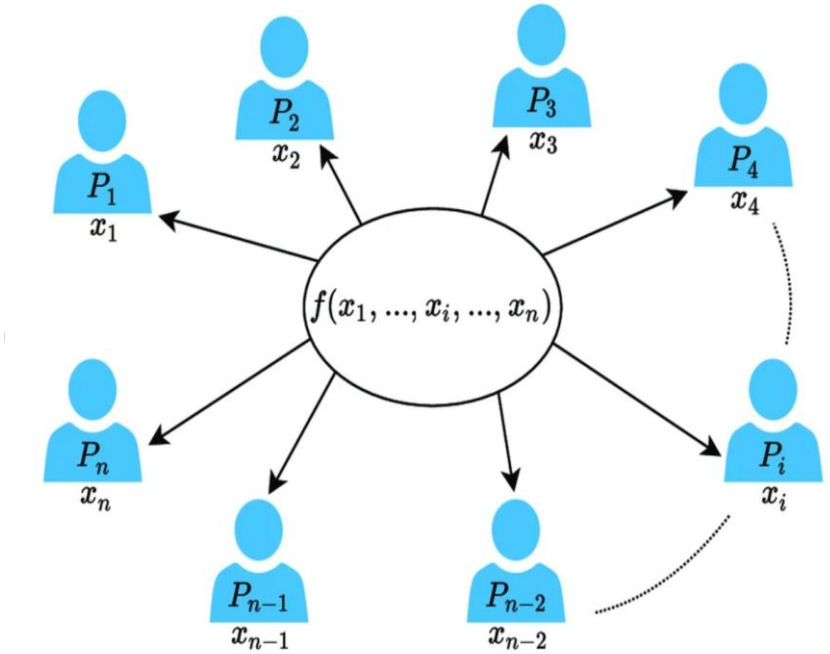
모델·입력을 n개 조각으로 분할
각 서버에 전달 — 조각만으론 복원 불가

② 조각끼리 연산 (평문 비공개)

덧셈·곱셈: 조각끼리 더하거나 곱해서 선형 연산 수행
비선형 함수(ex. ReLU): x>0인지 판단해야 하나,
조각만으로는 연산 불가

③ 결과 반환

각 서버의 결과 조각을 합산
사용자에게 최종 추론 결과 전달



MPC에서 x_i = Enc(모델 파라미터) 또는 Enc(사용자 입력)

→ 각 파티는 암호화된 조각만 보유, 원본 데이터는 끝까지 비공개

배경 지식: Threshold FHE (n-out-of-n)

핵심 아이디어

동형암호(FHE)의 비밀키를 MPC처럼 n개로 쪼개
복호화 자체를 여러 파티가 협력해서 수행

① 비밀키 s를 n개 조각으로 분산

$s = s_0 + s_1 + \dots + s_{n-1}$
각 P_i는 s_i만 보유, 전체 s를 아무도 모름

② 공개키 생성 (협력)

각 P_i: $p_{0,i} = -p_1 \cdot s_i + e_i$ 계산 후 공개
모든 파티가 $p_{0,i}$를 합산: $p_0 = \Sigma p_{0,i}$
→ 공개키 $pk = (p_0, p_1)$ 완성

③ n-out-of-n 분산 복호화

각 P_i: $h_i = c_1 \cdot s_i + e_i$ (부분 복호화)
P_0(or 임의의 파티)가 모든 h_i 수집 → $m = c_0 + \Sigma h_i$

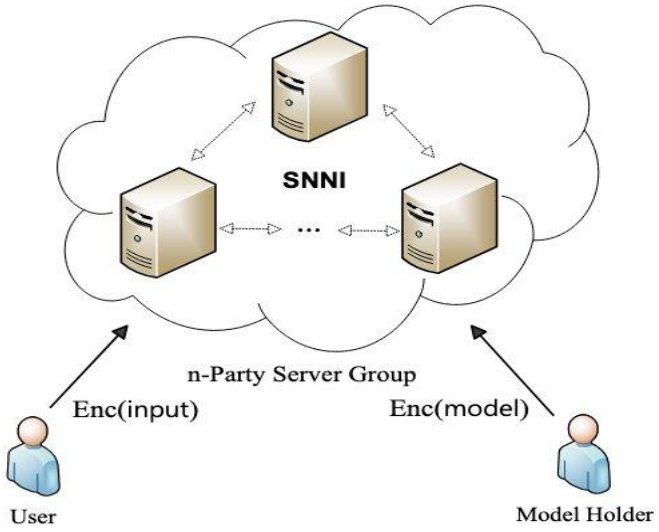


Fig. 3. TFMD system model.

단 1개 조각만 없어도 복호화 불가

→ n−1명이 결탁해도 원본 복호화 불가 (dishonest-majority)

TFMD = MPC의 협력 구조 + Threshold FHE의 분산 복호화 → 비선형 함수를 안전하게 계산

TFMD의 핵심 제안: Threshold FHE with Masked Data

비선형 함수(ReLU)를 안전하게 계산하려면 두 조건이 필요

Condition I: 키 보호

비밀키 s를 n개 조각(s_i)으로 분산
→ n-out-of-n threshold 복호화
→ 1개 조각만 없어도 복호화 불가
→ n−1명이 결탁해도 키 안전

Condition II: 입력 보호

랜덤값 r로 x를 마스킹
→ 복호화 결과는 r·x (x가 아님)
→ r을 평문으로 아무도 모름
→ r·x에서 x 역산 불가

두 조건을 결합 → Threshold FHE with Masked Data

키를 분산(Threshold FHE) + 입력을 랜덤값으로 마스킹(Masked Data)
→ 복호화 후에도 x를 알 수 없음 → 비선형 함수를 안전하게 계산

TFMD Framework Overview

Threshold FHE with Masked Data를 기반으로 구성된 4개의 프로토콜 + 완전 추론

B. 비선형 함수 프로토콜

ΠReLU

ReLU 안전 계산

`Enc(x) → Enc(Max(x, 0))`

랜덤값 r로 마스킹 → threshold 복호화 → r 제거 → ReLU 결과

ΠMP (Max Pooling)

Max Pooling 안전 계산

`Enc(x, y) → Enc(Max(x, y))`

ΠReLU와 동일한 방식, x vs 0 대신 x vs y 비교

C. 추론 프레임워크 (SNNI)

ΠEC (Encoding Conversion)

Coefficient → SIMD
선형↔비선형 연결

ΠFHE-PC (Parameter Conversion)

ParmA → ParmB
전처리↔온라인 연결

Complete Inference

최종 결과를 암호화된 채로
사용자에게 안전하게 전달

ΠReLU — 전처리 (Prep) 단계

① 각 P_i ($i \in [1,n-1]$) 개별 준비 후 P_1에게 전송

r_i, r'_i 생성 ($r_i \cdot r'_i = 1$) → $Enc(r_i)$, $Enc(|r'_i|)$, $Enc(Sign(r_i))$
→ P_1에게 전송

② P_1이 동형곱셈으로 합산 → P_0에게 전달

$$Enc(r) = \boxtimes Enc(r_i) = Enc(r_1) \boxtimes Enc(r_2) \boxtimes \dots \boxtimes Enc(r_{n-1}) = Enc(r_1 \cdot r_2 \cdot \dots \cdot r_{n-1})$$

→ r을 평문으로 아무도 모르지만, 암호화된 Enc(r)은 P_0가 갖게 됨 (마스킹용)

$$Enc(Max(r',-r')) = Enc(|r'|) = \boxtimes Enc(|r'_i|)$$

r ' 의 절댓값의 암호문
→ 실제 추론 실행 단계 3에서 마스킹된 r을 제거할 때 사용 ($r \cdot r' = 1$이므로)

$$Enc(Sign(r)) = \boxtimes Enc(Sign(r_i)) = Enc(Sign(r_1)) \dots \boxtimes Enc(Sign(r_{n-1}))$$
$$Enc(Min(0,r')) = \tfrac{1}{2} \boxtimes (Enc(Sign(r)) \boxminus Enc(1)) \boxtimes Enc(|r'|)$$

$r > 0$이면: $Sign(r)=1$ → $(1-1)/2 \times |r'| = 0$ → $Min(0,r')=0$ ✓
$r < 0$이면: $Sign(r)=-1$ → $(-1-1)/2 \times |r'| = -|r'| = r'$ → $Min(0,r')=r'$ ✓
→ r의 부호에 따라 0 또는 r'을 선택하는 스위치 (실제 추론 단계에서 사용)

입력 Enc(x)와 무관 → 미리 실행 가능 → 실제 추론 시 지연 감소

논문 Figure 4 — Prep 단계

Protocol Π_{ReLU}

Input: $Enc(x)$, P_i ($i \in [0, n-1]$) holds the key share s_i
Output: $Enc(Max(x, 0))$

Prep.:

1. P_i ($i \in [1, n-1]$) randomly generates $r_i, r'_i \in \mathbb{R}_q$, the decoded $r_i \cdot r'_i = 1$. Parties in $\mathcal{P}_{[1,n-1]}$ collaboratively calculate $Enc(r)$, $Enc(Max(r', -r'))$ and $Enc(Min(0, r'))$. P_1 sends the three ciphertexts to P_0.
   - Each P_i ($i \in [1, n-1]$) calculates $Enc(r_i)$, $Enc(|r'_i|)$, $Enc(Sign(r_i))$.
   - Each P_i ($i \in [2, n-1]$) sends $Enc(r_i)$, $Enc(|r'_i|)$, $Enc(Sign(r_i))$ to P_1.
   - $Enc(r)$: P_1 calculates $Enc(r) = \boxtimes_{i \in [1,n-1]} Enc(r_i)$.
   - $Enc(Max(r', -r'))$: P_1 calculates $Enc(Max(r', -r')) = Enc(|r'|) = \boxtimes_{i \in [1,n-1]} Enc(|r'_i|)$.
   - $Enc(Min(0, r'))$: P_1 first calculates $Enc(Sign(r)) = \boxtimes_{i \in [1,n-1]} Enc(Sign(r_i))$, then $Enc(Min(0, r')) = \frac{1}{2} \boxtimes (Enc(Sign(r)) \boxminus Enc(1)) \boxtimes Enc(|r'|)$.

Online:

1. P_0 calculates $Enc(r \cdot x) = Enc(r) \boxtimes Enc(x)$, sends $Enc(r \cdot x)$ to each P_i ($i \in [1, n-1]$).
2. Parties in $\mathcal{P}_{[0,n-1]}$ perform the n-out-of-n threshold decryption of $Enc(r \cdot x) = (c_0, c_1)$. P_0 obtains $r \cdot x$.
   - P_i ($i \in [1, n-1]$) calculates $h_i = c_1 \cdot s_i + e_i$, sends h_i to P_0.
   - P_0 calculates $h_0 = c_1 \cdot s_0 + e_0$, then calculates $c_0 + h_0 + \sum_{i=1}^{n-1} h_i$ to obtain $r \cdot x \in \mathbb{R}_q$.
3. P_0 calculates the output $Enc(Max(x, 0)) = (r \cdot x) \boxtimes Enc(Min(0, r')) \boxplus Max(r \cdot x, 0) \boxtimes Enc(Max(r', -r'))$.

Fig. 4. TFMD ReLU protocol Π_{ReLU}.

ΠReLU — 실제 추론 실행 (Online) 단계

Step 1 마스킹

P_0: $Enc(r \cdot x) = Enc(r) \boxtimes Enc(x)$

→ 전처리에서 받은 Enc(r)과 입력 Enc(x)를 동형곱셈 → 모든 P_i에게 전송

Step 2 n-out-of-n Threshold 복호화

각 P_i: $h_i = c_1 \cdot s_i + e_i$ → P_0에 전송 (부분 복호화)

P_0: $r \cdot x = c_0 + h_0 + h_1 + \ldots + h_{n-1}$

→ P_0가 평문 r·x 획득
→ r을 모르므로 x는 여전히 비밀 (Condition II 만족)

Step 3 ReLU 결과 계산 (P_0가 단독으로)

$Enc(Max(x,0)) = (r \cdot x) \boxtimes Enc(Min(0,r')) \boxplus Max(r \cdot x, 0) \boxtimes Enc(Max(r',-r'))$

※ Max(r·x, 0)은 P_0가 평문 r·x를 갖고 있으므로 직접 계산 가능

4가지 경우:
- x>0, r>0 → r·x>0 → Max(r·x,0)=r·x, Min(0,r')=0
  → (r·x)×0 + r·x×r' = r·x·r' = x → 결과: **x**
- x>0, r<0 → r·x<0 → Max(r·x,0)=0 → 결과: **x**
- x≤0, r>0 → r·x≤0 → Max(r·x,0)=0 → 결과: **0**
- x≤0, r<0 → r·x≥0 → Max(r·x,0)=r·x → 결과: **0**

→ x를 아무도 평문으로 보지 않고 Enc(Max(x,0)) 출력

논문 Figure 4 — Online 단계

Protocol Π_{ReLU}

Input: $Enc(x)$, P_i $(i \in [0, n-1])$ holds the key share s_i
Output: $Enc(Max(x, 0))$
Prep.:

1. P_i $(i \in [1, n-1])$ randomly generates $r_i, r'_i \in \mathbb{R}_q$, the decoded $r_i \cdot r'_i = 1$. Parties in $\mathcal{P}_{[1,n-1]}$ collaboratively calculate $Enc(r)$, $Enc(Max(r', -r'))$ and $Enc(Min(0, r'))$. P_1 sends the three ciphertexts to P_0.
   - Each P_i $(i \in [1, n-1])$ calculates $Enc(r_i)$, $Enc(|r'_i|)$, $Enc(Sign(r_i))$.
   - Each P_i $(i \in [2, n-1])$ sends $Enc(r_i)$, $Enc(|r'_i|)$, $Enc(Sign(r_i))$ to P_1.
   - $Enc(r)$: P_1 calculates $Enc(r) = \boxtimes_{i \in [1,n-1]} Enc(r_i)$.
   - $Enc(Max(r', -r'))$: P_1 calculates $Enc(Max(r', -r')) = Enc(|r'|) = \boxtimes_{i \in [1,n-1]} Enc(|r'_i|)$.
   - $Enc(Min(0, r'))$: P_1 first calculates $Enc(Sign(r)) = \boxtimes_{i \in [1,n-1]} Enc(Sign(r_i))$, then $Enc(Min(0, r')) = \frac{1}{2} \boxtimes (Enc(Sign(r)) \boxplus Enc(1)) \boxtimes Enc(|r'|)$

Online:

1. P_0 calculates $Enc(r \cdot x) = Enc(r) \boxtimes Enc(x)$, sends $Enc(r \cdot x)$ to each P_i $(i \in [1, n-1])$.
2. Parties in $\mathcal{P}_{[0,n-1]}$ perform the n-out-of-n threshold decryption of $Enc(r \cdot x) = (c_0, c_1)$. P_0 obtains $r \cdot x$.
   - P_i $(i \in [1, n-1])$ calculates $h_i = c_1 \cdot s_i + e_i$, sends h_i to P_0.
   - P_0 calculates $h_0 = c_1 \cdot s_0 + e_0$, then calculates $c_0 + h_0 + \sum_{i=1}^{n-1} h_i$ to obtain $r \cdot x \in \mathbb{R}_q$.
3. P_0 calculates the output $Enc(Max(x, 0)) = (r \cdot x) \boxtimes Enc(Min(0, r')) \boxplus Max(r \cdot x, 0) \boxtimes Enc(Max(r', -r'))$.

Fig. 4. TFMD ReLU protocol Π_{ReLU}.

프레임워크 나머지 구성 요소

ΠMP — Max Pooling

ReLU와 동일한 방식, 단 x vs 0 → x vs y
같은 r이 x, y 모두에 곱해지므로 대소관계 보존
Enc(r), Enc(r'), Enc(Sign(r)) 사전 준비

ΠEC — Encoding Conversion

Coefficient 인코딩 → SIMD 인코딩 변환
랜덤값 a로 마스킹 후 복호화·재인코딩
(ΠReLU와 동일한 masked data 기법)

Parameter Conversion (ΠFHE-PC)

전처리: ParmA (N=8192, $q \approx 2^{218}$) — 동형곱셈 많음
온라인: ParmB (N=4096, $q \approx 2^{109}$) — 빠른 실행
ΠEC와 동일한 기법으로 파라미터 전환

결론

핵심 기여

Threshold FHE with Masked Data

비선형 함수(ReLU/Max Pooling)를 암호화된 채로 안전하게 계산

보안

임의의 n명 지원

프로토콜 재설계 불필요

최대 n−1명 악의적 참여자

기존: t < n/2 (honest-majority)
TFMD: 단 1명만 정직해도 안전 (dishonest-majority)

성능

4.9× ReLU

6.7× Max Pooling

2.1× 추론

vs CrypTFlow (WAN, 3-party)

한계

Semi-honest only — 프로토콜을 따르되 정보를 모으는 적대자만 방어

향후

악의적 적대자 대응 | GPU 가속 통합