비밀 프로그램 정적 분석

이우석, 허기홍, 이광근

서울대학교

{wslee, khheo, kwang}@ropas.snu.ac.kr

하고 싶은 것

• 프로그래머는 자신의 프로그램, 분석서비스 제공자는 분석기를 서로에게 조금도 노출시키고 싶지 않음

• 양측의 완벽한 보안을 지키면서 분석서비스 제공

배경지식

공개키 암호체계란? 암호는 아무나, 복호화는 특정인만 비밀키로

확률적 암호? 같은 평문이라도 암호화 할 때마다 결과다름(통계 공격 회피)

동형암호(Homomorphic Encryption)? 암호문에 연산 적용한 결과와 평문에 연산 적용한 결과를 암호화한 것이 같은 연산을 지원하는 암호체계

$$\underline{op}(\mathcal{E}(m)) \equiv \mathcal{E}(op(m))$$

완전동형암호(Fully Homomorphic Encryption)? 모든 종류의 연산을 지원하는 동형암호

$$\mathcal{M} = \mathbb{Z}_2 = \{0, 1\}$$

$$\mathcal{E}(m_1) + \mathcal{E}(m_2) \equiv \mathcal{E}(m_1 + m_2)$$
$$\mathcal{E}(m_1) \times \mathcal{E}(m_2) \equiv \mathcal{E}(m_1 \times m_2)$$

덧셈은 XOR, 곱셈은 AND에 해당, 이 둘을 이용해서 모든 회로를 시뮬레이션 할 수 있음

완전동형암호의 간단한 예 (공개키는 pq, 비밀키는 p) :

$$\mathcal{E}(m) = m + pq + 2\epsilon$$
$$\mathcal{D}(c) = (c \bmod p) \bmod 2$$

• p보다 작은 난수
• 같은 평문이 똑같은 암호문이 되지 않게 하는 역할.
• 노이즈 누적의 원인

<u>완전동형암호는 실용성이 문제!</u>

• 동형적 연산을 거듭하면 노이즈가 커짐 **(곱셈(AND)이 노이즈 크게 키움)**

• 노이즈가 비밀키보다 커지면 복호화가 불능

• 이를 해결하는 비용이 비쌈. 그러므로 노이즈 조절이 관건

간단한 해결책과 문제

간단한 방법 : 암호화된 프로그램을 동형적으로 분석

$$\underline{\mathcal{A}}(\mathcal{E}(P)) \equiv \mathcal{E}(\mathcal{A}(P))$$

안되는 이유 :

• 프로그램이 암호화됨에 따라 생기는 비효율

예) <u>실행함수</u> (<u>명령문</u>) =

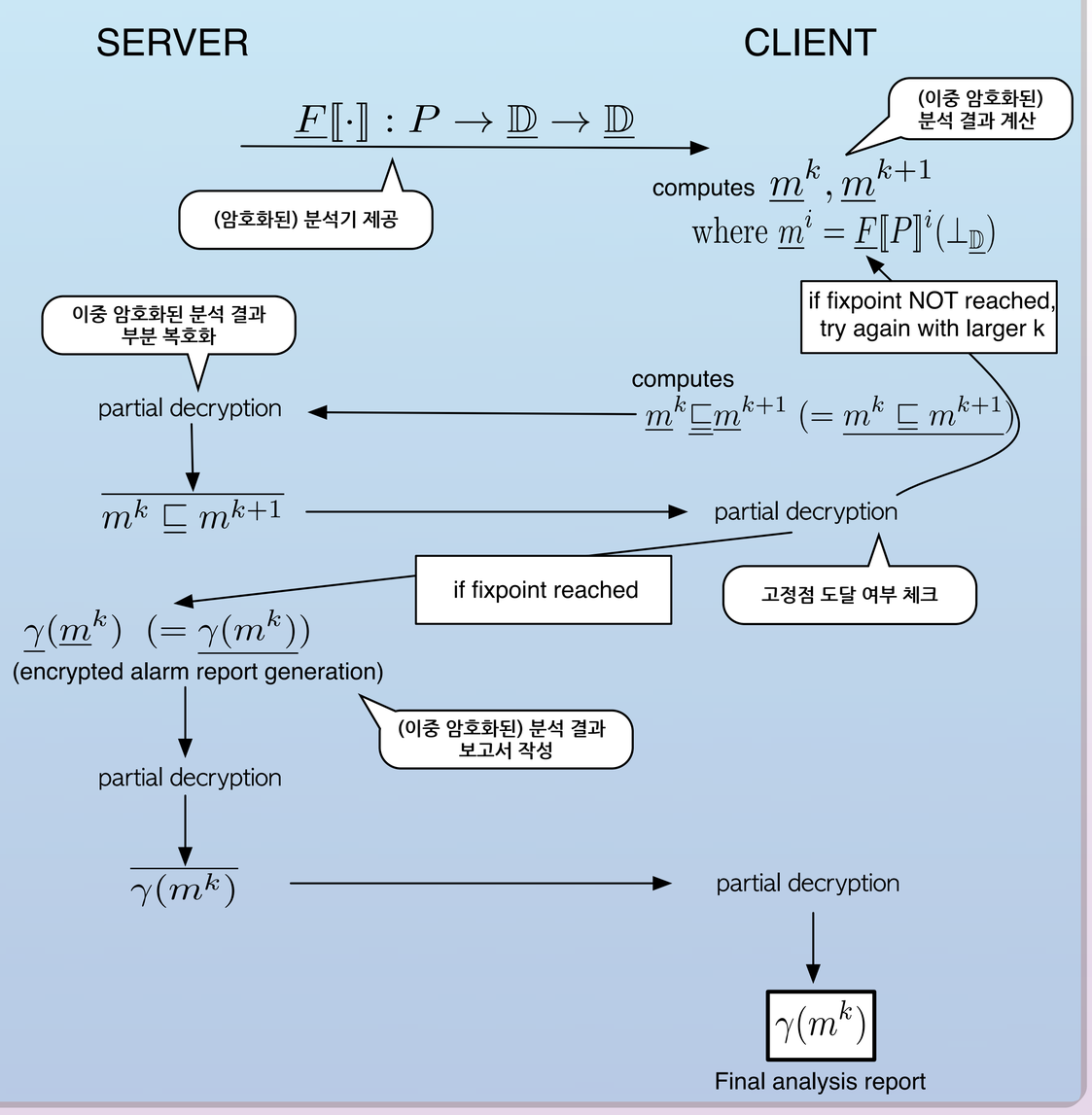
(명령문 ≟ 할당문) × (할당문의 좌변 ≟ 변수1) × (<u>실행결과1</u>) +
(명령문 ≟ 할당문) × (할당문의 좌변 ≟ 변수2) × (<u>실행결과2</u>) +
……
(명령문 ≟ 분기문) × (분기문의 형태 ≟ 부정 조건문) × (<u>실행결과3</u>) +
(명령문 ≟ 분기문) × (분기문의 형태 ≟ 조건문) × (<u>실행결과4</u>)
……

해결방안

1. 암호화된 분석기를 클라이언트가 실행시킴
   (프로그램 암호화할 필요 없음)

2. 분석 과정에서 노이즈가 누적되지 않게 조절

프로토콜



노이즈 누적을 회피하는 법

요약전이함수를 거듭 적용하는 것을 피하기(유한도메인만 가능):

• 요약전이함수가 내재한 노이즈가 누적되지 않게

단계1. 다음 테이블 구성	단계2. 지금까지 계산된 값과 같은 원소를 동형적으로 찾기 (처음엔 $\underline{X} = \underline{F}(\bot)$)	단계 3.
$(d_1, \ldots, d_m \in \mathbb{D})$	$\underline{X}$	값이 고정점이면 완료
$\underline{d_1}$ | $\underline{F}(\underline{d_1})$	$\underline{d_1}$ | $\underline{F}(\underline{d_1})$	아니면 <u>X</u>를 갱신 후 단계 2로
… | …	… | …	
$\underline{d_m}$ | $\underline{F}(\underline{d_m})$	$\underline{d_m}$ | $\underline{F}(\underline{d_m})$	

• 동형적 동일성 검사는 노이즈를 거의 유발하지 않음.

• 전체 노이즈는 F가 유발하는 노이즈 S (원시적인 방법 : S^n)

• $\forall x, y. F(x) \sqcup F(y) = F(x \sqcup y)$ 일 때 : 테이블의 엔트리 수 감소

• $F : 2^D \to 2^D$ 일때, 2^D 대신 D 의 원소들에 대해서만 테이블 구성

• 동일성 검사 대신에 소속여부 검사

앞으로

• 더 복잡한 도메인에 대해서 노이즈 누적을 회피할 방법 모색

• 적절한 인스턴스 찾아서 구현

• 안드로이드 앱의 개인정보누출 분석기를 고려 중