김 경 진 교수 지도

석사학위 청구논문

크리덴셜 스터핑 공격 대응에 활용하는 Large Asymmetric PSI 연산 기법

2024

성신여자대학교 대학원

미래융합기술공학과

윤 지 희

크리덴셜 스터핑 공격 대응에 활용하는 Large Asymmetric PSI 연산 기법

김 경 진 교수 지도

이 논문을 석사학위논문으로 제출함

2024년 05월

성신여자대학교 대학원
미래융합기술공학과
윤 지 희

인 준 서

윤지희의 석사학위 논문으로 인준함

2024년 06월

심사위원장 김 성 민 (서명 또는 인)

심 사 위 원 김 경 진 (서명 또는 인)

심 사 위 원 이 주 희 (서명 또는 인)

성신여자대학교 대학원

논 문 개 요

C3(Compromised Credential Checking) 서비스는 탈취한 계정정보를 무작위로 입력해 접근 권한을 획득하는 크리덴셜 스터핑(Credential Stuffing) 공격 대응방안이다. 이는 서비스 제공자가 유출된 자격증명을 수집해 DB를 구성하고, 사용자는 서버에 사용자 계정과의 비교 연산을 요청해 자격증명 유출 여부를 확인하는 서비스로, 두 당사자가 보유한 개인정보 데이터 간의 교집합을 얻고자 하는 연산 상황에 PSI(Private Set Intersection) 프로토콜을 이용한다.

기존 비대칭 PSI 프로토콜은 비대칭 데이터 셋 연산 환경에서 중복 연산을 수행하며 효율성과 안전성 향상에 목적을 가졌다. 그러나 유통되는 탈취계정의 증가는 C3 서버 데이터 증가에 따라 다회성의 중복 연산을 요구한다. 이에 중복 연산으로 인한 연산 및 통신 복잡도 상쇄 방안이 요구되며, 보다 대량의 데이터 연산에서도 효율성을 가져야 한다. 따라서 본 연구에서는 대량의 데이터가 업데이트되는 비대칭 PSI 연산 과정에서 효율성을 높일 수 있는 Large Asymmetric PSI 연산 방안을 제안 및 증명한다. 수신 노드가 $Y=2^{10}$의 데이터를, 발신 노드는 기존의 세 가지 비대칭 환경인 $X=2^{20}$, 2^{22}, 2^{24}의 데이터를 보유하되, 각각 2^{14}, 2^{16}, 2^{18} 만큼의 데이터를 추가하여 다회 연산 수행 시, 결과적으로 X의 양이 대량일수록 Large Asymmetric PSI의 연산 효율을 더 크게 가지고, 연산 시간 측면에서 tag 생성 및 복구 소요시간을 상쇄시키는 것을 증명하였다.

목 차

표 차 례

그림 차례

Ⅰ. 서 론

1. 연구 배경

정보통신기술의 발달과 함께 다양한 웹·앱 서비스가 등장하였으며, 대다수 서비스 제공자는 많은 사용자를 관리하기 위해 아이디와 패스워드 기반의 인증을 제공한다. 사용자는 간편하게 로그인을 통해 접근 권한을 획득하나, 대부분 사용자는 더 많은 서비스에 가입할수록 기억하기 쉽게 같거나 유사한 암호를 선택한다[1,2,3]. 이에 따라 공격자에게도 사용자 인증 정보는 쉬운 접근 권한의 획득 수단이 되었고, 피싱, 악성코드 등의 방법으로 손쉽게 계정을 탈취하는 공격이 성행하고 있다. 취득한 계정정보는 공격에 활용하거나, 다크웹 등의 경로를 통해 판매하여 크리덴셜 스터핑(Credential stuffing)[4], 크리덴셜 트윅(Credential Tweak) 공격에 이용되고 있다. 크리덴셜 스터핑 공격을 통해 1,000회 시도에서 60%의 계정을 성공적으로 취득할 수 있고[5], 누출된 암호에 작은 조정값을 적용해 성공 확률을 높이는 크리덴셜 트윅은 1,000회 이하의 변형 시도로 83%의 계정을 성공적으로 침해할 수 있어[6] 추가 피해를 일으킨다.

계정 탈취 공격 피해에 대응하기 위한 수단으로 유출된 자격증명을 수집해 DB를 구성하고, 사용자가 직접 계정과의 비교 연산하여 자격증명 유출 여부를 확인하는 서비스를 C3(Compromised Credential Checking)라고 한다. 이는 두 당사자가 보유한 개인정보 데이터 셋 간의 교집합을 얻고자 하는 연산 상황에 비대칭 PSI(Asymmetric Private Set Intersection) 프로토콜을 이용한다. PSI(Private Set Intersection)는 상호작용 기반의 암호 프로토콜로 두 데이터 셋을 입력값으로 그들의 교집합을 찾고, 연산에 참여

한 한 당사자 혹은 두 당사자 모두에게 교집합 값을 출력하는 연산이다. 출력값을 얻는 소량의 데이터를 보유한 한쪽 당사자를 수신 노드(Receiver), 대량의 데이터를 보유한 나머지 한쪽 당사자를 발신 노드(Sender)라고 한다. 이는 왓츠앱(WhatsApp)과 같은 모바일 메신저에서 개인 연락처를 검색하거나[7], 전염병 확산을 제어하기 위해 접촉추적[8], 원격진단[9], 온라인 광고 효과 측정[10]에 활용되는 등 다양한 활용 분야에 이용되고 있다.

C3는 발신 노드의 집합이 지속해서 최신화되고 있으며, 수신 노드의 집합 크기가 매우 작고 발신 노드 집합이 매우 큰 상황인 비대칭한(Asymmetric) 환경을 가진다. 이에 따라 기존의 비대칭 PSI 프로토콜은 수신 노드의 인풋 정보 일부를 이용해 대량의 데이터를 보유한 서버에서 연관 데이터를 추출한 후 비교 연산을 수행한다. 이 과정에서 사용자의 일부 정보가 서버에 노출됨에 따라 계정 탈취의 위험성이 존재한다. 또한, 데이터가 고정된 일회성 연산의 효율성을 높이기 위한 연구가 주로 이루어지고 있다. 이에 기존의 C3의 낭비되는 통신 복잡도를 상쇄할 방안이 필요하다. 따라서 본 연구에서는 크리덴셜 스터핑 공격에 사용자가 능동적으로 대응할 수 있는 수단인 C3에서 대량의 유출 계정을 업데이트하는 상황을 고려하여 다회성 연산이 가능한 Large Asymmetric PSI 연산기법을 연구한다.

2. 연구 목적 및 기여점

C3는 유출된 자격증명 집합에 사용자 계정의 유무를 확인하여 유출 여부를 확인하는 서비스로, 계정정보를 안전하게 관리해야 하는 서비스 제공자와 2차 피해에 대응해야 하는 서비스 이용자 모두에게 이점을 제공한다. 특히 공격자는 침해사고 사례 초기 진입 단계에 크리덴셜 스터핑을 높은 빈도로 이용하고 있으며, 다량의 계정 유출 사례가 빈번하게 이루어지므로 C3를 통한 빠르고 쉬운 탐지가 필요하다.

그러나 정적인 데이터 환경의 비대칭 PSI 프로토콜을 이용하는 C3는 발신 노드의 데이터 증가와 수신 노드의 반복 연산 요청 시 연산 복잡도를 대폭 증대시킨다. 따라서 동적인 데이터 환경에서 적용할 수 있는 Large Asymmetric PSI 연산기법을 통해 대량의 데이터와의 효율적인 연산이 가능하고, 계정정보에 대한 안전성을 유지하는 방안을 연구한다.

본 논문의 주요 기여점은 다음과 같다.

첫째, 증가하는 계정 유출 사례 및 크리덴셜 스터핑 공격에 대응하기 위한 C3의 종래기술을 분석하여 한계점을 도출하였다. 기존의 C3 서비스의 종래기술과 비대칭 PSI 연산 및 업데이트 가능한 PSI를 분석하여 한계점을 도출하고 개선방안을 분석하였다.

둘째, 완전동형암호화 기반의 PSI 프로토콜로 종래 방식과 제안 방식을 구현하여 서버의 데이터 보유량의 변화에 따른 전체 통신량, 수신 노드의 발신 노드에의 전송량, 발신 노드의 수신 노드에의 전송량, 발신 노드의 전처리 시간 및 *tag* 생성 시간을 평가하여 효율성을 증명하였다.

셋째, 다량의 데이터가 빈번하게 업데이트되는 환경에서 이용 가능한 안전하고 효율적인 Large Asymmetric PSI 연산 방안을 제안하였다. 특히, 다

량의 계정 유출 사례를 신속하고 효율적으로 탐지하기 위한 C3의 개선 방안을 도출하고 기존 static PSI의 연산 복잡도 문제 해결을 위해 동적인 데이터 환경에서의 효율성을 높이는 연산 방안을 제안하였다.

본 논문의 구성은 다음과 같다. 2장은 C3와 관련된 PSI 프로토콜의 관련 연구를 소개하고, 3장에서는 2장을 바탕으로 PSI 연산의 한계점을 분석한다. 4장에서는 본 논문에서 제안하는 Large Asymmetric PSI 연산기법으로 문제점 해결 방안을 설명한다. 5장에서는 실험 환경을 소개하고, 실험에 사용한 파라미터별 비교지표에 따른 실험결과를 분석한다. 6장에서는 본 연구의 결론으로 논문을 마무리한다.

Ⅱ. 관련 연구

본 장에서는 연구의 배경이 되는 PSI 프로토콜과 동형암호 기반의 PSI 프로토콜, 메시지 기반 암호화와 C3의 개념을 소개하고, 이의 연구 동향을 파악해 한계점을 분석한다. 특히 수신 노드와 발신 노드 간 비대칭 환경에서의 종래기술과 선행연구를 분석하여 한계점을 도출하고, 이의 개선 방향성을 도출한다.

1. 관련 기술

1) APSI(Asymmetric PSI)

PSI는 두 당사자 각각이 개인 집합을 보유하고 있을 때, 교집합을 제외한 어떠한 정보도 공개하지 않으면서 각 집합의 교집합을 알고 싶어 하는 상황에 이용할 수 있다. 첫 번째 개념은 Diffie-Hellman 키 교환을 기반[11]으로 제안하였으며, 연산 속도 향상을 위한 방향으로 발전한다. 기본적인 통신 복잡도는 연산 집합의 크기에 선형적으로 증가하나, 한쪽의 집합이 다른 것보다 상당히 작은 비대칭 연산은 프라이버시를 고려한 연산의 경우 큰 부하가 요구된다. 이에 비대칭 데이터 환경에 최적화된 연산 방안이 등장하였다. 기존 PSI 연산 4가지를 데이터 크기가 비대칭한 환경에서도 효율성을 가질 수 있도록 Bloom 필터를 이용해 연산 부담 및 저장공간의 부담을 줄여주는 방식을 통해 개선하고, 서버의 입력 집합을 줄이는 연산 후 축소된 집합에서 전통적인 PSI 연산을 수행한 연구[12]가 있다. 그러나 서버의 안전성을 보장이 필요하다. 이처럼 일정 오버헤드를 가지더라도 안전성을 높이는 방식과 효율성을 개선하는 방식으로 발전하고 있다.

2) 동형암호(Homomorphic Encryption) 기반 APSI

동형암호는 암호화 데이터를 복호화 없이 연산할 수 있는 기술로 개념적으로 1978년에 처음 소개되었으며, 2009년 Craig Gentry가 암호화한 상태에서 임의의 연산을 무한 반복할 수 있는 부트스트래핑(Bootstrapping) 기법을 활용한 완전 동형암호(Fully Homomorphic Encryption)를 제안하였다. 현재 완전 동형암호는 젠트리의 1세대 스킴(Scheme)을 시작으로, 최초의 사용 가능한 2세대 BGV, BFV, 작은 데이터 처리에 효과적인 3세대 CGGI, 실수 연산이 가능한 4세대 CKKS2까지 개발되었으며, 본 연구에서는 BFV 스킴을 통한 연산 방안을 이용한다.

동형암호의 성질을 이용해 제3자에게 암호화 데이터를 전달하여 데이터를 보호하면서 분석을 수행할 수 있다. 평문 M_1, M_2에 대해 동형암호화하였을 때, $C_1 = Enc(M_1)$, $C_2 = Enc(M_2)$라하고, 간단한 연산을 수행하였을 때, 두 평문의 합은 $Dec(Enc(M_1) + Enc(M_2)) = M_1 + M_2$와 같이 암호문을 더하고 이를 복호화한 값과 같으며, 두 평문의 곱은 $Dec(Enc(M_1) * Enc(M_2)) = M_1 * M_2$와 같이 암호문을 곱하고 이를 복호화한 값과 같다.

동형암호 기반의 PSI 연산 과정은 FIGURE 1과 같다. 비대칭 PSI 프로토콜은 양자가 α-bit의 문자열 집합을 보유하고 있으며, 발신 노드는 N_x크기의 데이터 X를, 수신자는 N_y 크기의 데이터 Y를 보유할 때, $N_x > N_y$ 로 수신자가 모바일 장치와 같은 계산능력이 제한된 경우에 유용하게 동작한다. 수신자가 Y집합을 암호화하여 발신 노드에 보내고, 발신 노드는 Y집합에서 X집합의 원소를 제거하는 비교 연산을 수행하여 출력을 수신 노드에 보낸다. 수신 노드는 복호화를 통해 교집합의 원소를 확인한다. 이때, 연산 결과가 0인 경우 $Y \in X$이며, 0이 아닌 경우 $Y \notin X$이다. 결과적으로 연산 결과 발신 노드에는 아무것도 노출하지 않고 수신 노드만 $X \cap Y$를 확인할 수 있다.

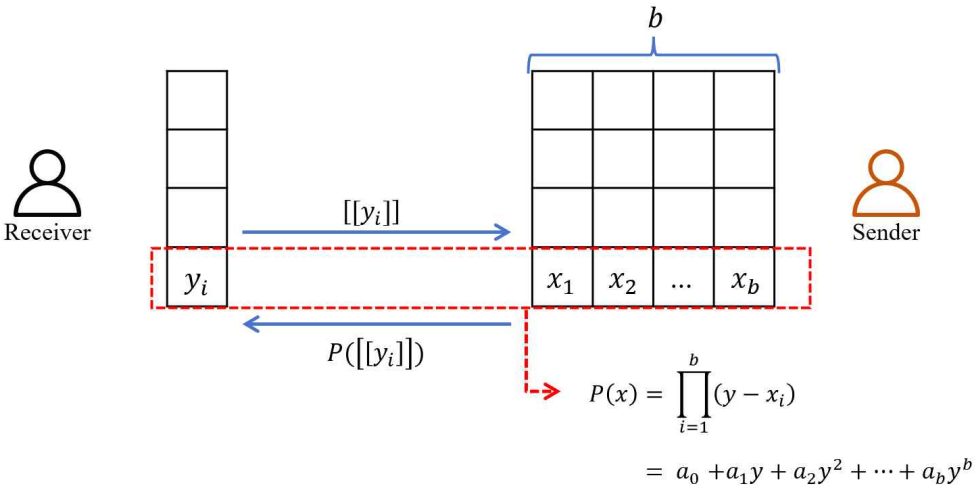

FIGURE 1. 동형암호 기반 PSI 연산 매커니즘

3) 메시지 기반 암호화

데이터 중복 처리 기술은 두 개 이상의 중복데이터가 있으면 하나만 저장하고 나머지는 포인터로 대체하여 같은 데이터를 반복해서 저장하지 않고 중복되는 부분을 제거하는 기술이다. 이는 스토리지 비용 감소, 데이터 백업 시간 단축, 전송량 감소 등 IT 비용 절약을 위해 사용된다. 서버나 제3자에 의해 저장된 데이터가 노출되는 것을 방지하기 위해 클라이언트와 서버에서 암호 데이터에 대한 중복 처리 기술이 필요하다.

중복제거가 가능한 같은 평문이라도 이를 암호화하려고 하는 사용자마다 서로 다른 비밀키를 가진다면, 하나의 평문에 대해 다수개의 서로 다른 암호문이 생성된다. 따라서 일반적인 대칭키 암호 방식으로는 다른 사용자가 암호화한 암호 데이터의 중복제거는 불가능하다. 이러한 문제를 해결하기 위해 사용자마다 다른 비밀키를 사용하지 않고, 데이터에 의해 키가 결정되는 수렴 암호화(Convergent Encryption) 방식[13]이 제안되었으며, 데이터 무결성 보존을 위해 tag를 생성하는 방안[14]이 제안되었다. FIGURE 2

은 암호화 프로세스 알고리즘을 도식화하여 나타냈다.

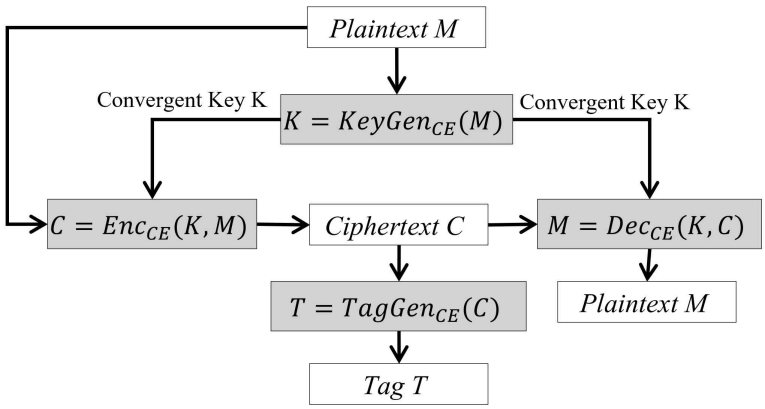

FIGURE 2. 수렴 암호화 매커니즘

우선 $KeyGen_{CE}(M) \rightarrow K$는 데이터 M을 기반으로 수렴키 K를 생성한다. 수렴 암호화를 안전하게 사용하기 위해서 수렴키는 메시지 인증코드(Message Authentication Code, MAC)를 이용한 예측 불가능성을 가져야 하는데, 이를 위해 $Hash(M) \rightarrow H_b$와 같이 데이터 M을 해시하여 H_b를 생성하고, 이와 무작위 매개변수 secret 값을 이용해 $KeyHmac(\mathrm{sec}ret, H_b) \rightarrow K$와 같이 무작위 수렴키 K를 생성할 수 있다.

$Enc_{CE}(K, M) \rightarrow C$는 위에서 생성한 키와 메시지 M을 대칭키 알고리즘으로 암호화하여 암호문을 출력한다. 해당 암호문은 $Dec_{CE}(K, C) \rightarrow M$와 같이 같은 키값으로 복호화하여 메시지를 복구한다. 암호문 값을 노출하지 않고 중복 데이터 제거에 이를 이용하기 위해서 $TagGen_{CE}(C) \rightarrow T_C$와 같이 태그값을 생성한다. 암호문 C를 해시 암호화하여 태그 T_C를 생성할 수 있다.

4) C3

C3는 Compromised Credential Checking으로, 유출된 자격증명 집합에 본인의 데이터 존재 여부를 확인하는 방법이다. C3는 사용자와 기업이 크리덴셜 스터핑 및 크리덴셜 트윅 공격을 완화할 수 있도록 한다[1,5,15]. FIGURE 3은 C3의 시스템 모델이며, C3 서버는 침해 데이터베이스에 액세스하여 서비스를 제공한다. n 크기의 데이터베이스 D는 패스워드 집합 $\{W_1,...,W_n\}$ 또는 아이디-패스워드 쌍 $\{(U_1,W_1),...,(U_n,W_n)\}$ 중 하나로 구성될 수 있다. 이용자는 자격증명 $S=(U,W)$을 입력으로 사용하고 노출 여부를 확인한다. 따라서 클라이언트와 서버는 $s \in D$ 인지 여부를 결정하기 위한 프로토콜을 수행한다. 여기서 클라이언트는 브라우저 확장 프로그램을 사용하여 C3에 쿼리하거나, 다른 웹 서비스를 통해 C3에 쿼리할 수 있다. 클라이언트는 C3에 다회 쿼리할 수 있지만, 횟수는 제한될 수 있다. 본 연구에서는 특히 C3를 집중적으로 분석하여 발견되는 문제점 개선 방안을 도출한다.

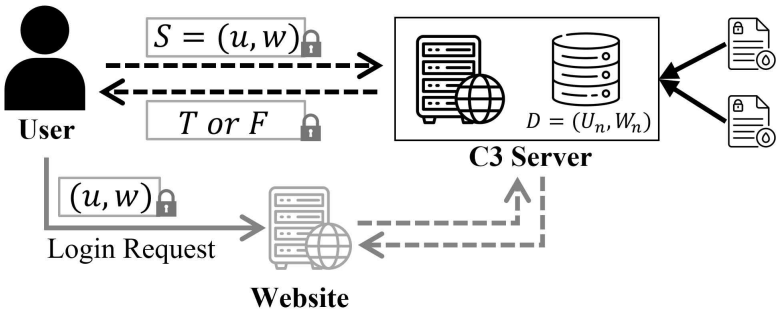

FIGURE 3. C3 매커니즘

2. C3 종래기술 분석

C3는 이용자가 직접 계정정보를 입력하여 기업이 보유한 유출된 계정 DB와 비교 연산을 수행해 유출 여부를 알려주거나, 일부 기업에서는 클라이언트 브라우저와 로그인 서버 백엔드에서 서비스한다. 이를 통해 사용자 계정의 침해 여부를 사전에 확인할 수 있다. C3 서비스의 PSI 기능을 TABLE I로 정리하였다.

Have I Been Pwned (HIBP)[16] 서비스는 최초의 C3로 Firefox 등 일부 웹 브라우저에서 사용된다. k-익명성 및 SHA-1을 이용한 연산 과정에서 클라이언트는 패스워드 hash-prefix 값 20bit를 C3 서버로 전송한다. C3 서버는 같은 20 bit hash-prefix를 가진 값을 클라이언트에게 응답하고, 클라이언트는 서버가 반환한 집합과 비교하여 탈취 여부를 확인한다. 그러나 HIBP는 공격자가 서버에서 반환된 해시값을 통해 무차별 대입 공격 통해 패스워드를 복구할 수 있는 문제점을 가진다[17]. 또한, 서버 보유 데이터양의 증가는 통신량과 연산 복잡도를 대폭 상승시킨다. ENZOIC[18]은 HIBP와 유사하지만, 사용자의 익명성을 보호하는 수단으로 클라이언트는 아이디 또는 아이디 U의 해시를 질의하고 솔트값 s를 받는다. 다음 라운드에서 클라이언트는 자신의 패스워드 P 로 $H(U,P,s)$를 계산하고 이 해시의 처음 40bit를 ENZOIC에 보내 모든 일치하는 값을 반환받아 연산을 수행한다. 그러나, 서버의 보유 데이터양의 증가는 서버의 통신량과 연산 복잡도를 상승시킨다. Password Checkup[19]은 Chrome 브라우저 확장 프로그램으로 사용자 계정의 탈취 여부를 확인하는 서비스를 제공한다. 연산 과정에서 서버가 보유한 유출 계정의 ID-Password 쌍을 해싱하고, 서버 비밀키로 암호화된다. 이후, 암호문을 포함하는 데이터베이스는 2바이트 hash-prefix를 버킷 ID로 이용해 분할된다. 이러한 Password

Checkup[20, 21]은 서버가 수십억 건의 손상된 자격증명을 보유하므로 엄청난 대역폭이 필요하다. 지속적인 서버 데이터의 증가에 따른 영향을 축소할 방안이 없다. 또한, 패스워드 정보를 포함한 hash-prefix를 C3 서버에 전송할 때, 서버가 손상된 상황에서 공격자가 패스워드를 더 쉽게 추측할 수 있는 문제점을 가진다. Microsoft는 웹 브라우저 Edge에서 사용자 계정 보호를 위한 Password Monitor[22]를 제공한다. 이 서비스는 동형암호(Homomorphic Encryption) 기반의 PSI 프로토콜[23, 24]을 이용한다. 클라이언트와 서버는 먼저 유효한 사전공격을 방지하기 위해 OPRF 프로토콜을 실행하여 질의할 자격증명의 해시값을 얻는다. 이후, 클라이언트는 해시값을 동형암호화하고 암호화된 쿼리를 C3 서버로 전송한다. 서버는 수신한 암호문에 대한 비교 연산을 수행하고 클라이언트에 암호화된 답변을 출력한다. 클라이언트는 답변을 복호화하여 True 또는 False의 최종 답변을 추출한다. Password Monitor는 사용자 정보의 노출을 제어하기 위해 동형암호를 적용해 안전성을 확보한다. 그러나, 지속적인 발신 노드의 보유 데이터 증가는 연산 복잡도를 증가시키고, 일치함수 평가의 복잡성을 높인다[25].

TABLE I. C3 종래기술 분석

ref	기여점	한계점
[16, 17]	• 최초의 C3로 k-익명성 및 SHA-1 연산 • 아이디, 패스워드를 구분해 쿼리 받으므로 서버가	• 사전공격, 무작위 대입 공격을 통한 패스워드 복구 가능 • false-positive 가능성

	입력값의 연관성 파악 방지	
[18]	• SHA-1과 솔트값을 이용해 사용자 쿼리 보호	• 인풋 데이터의 일부를 질의하고 솔트값 수신하는 과정에서 데이터 노출 가능성 • 서버의 보유 데이터양의 증가는 서버의 통신량과 연산 복잡도를 상승시킬 수 있는 문제
[19, 20, 21]	• Password Checkup은 Diffie-Hellman 프로토콜을 이용해 연산을 수행함	• 서버 대역폭의 막대한 증가 • Hash-prefix 값의 노출로 인한 패스워드 유추 문제
[22, 23, 24]	• Password Monitor는 연산 효율성 및 안전성을 증대하기 위해 C3 최초로 연산 과정에서 OPRF와 동형암호를 이용함	• 지속적인 발신 노드 데이터 업데이트 시 연산 부담 증가 • 평가함수의 복잡성

이처럼 종래기술은 사용자 계정 보호를 위해 C3를 제공하고 있으나, 해당 서비스 이용 시 계정정보 일부를 누출해 공격 위험성을 완전하게 개선하지 못했다. 연산 속도와 데이터 안전성과의 trade off 문제를 가지며, 발신 노드의 데이터양이 증가함에 따라 연산 복잡성의 증가로 연산 속도의 지연이 불가피한 문제점을 가지고 있다. 따라서 지속해서 업데이트되는 데이터를 고려하여 안전성을 유지하는 연산 방안이 필요하다.

3. C3 선행연구 분석

비대칭 데이터 환경에서의 연산을 수행하는 C3는 hash-prefix로 인한 관련 정보 노출 문제를 가진다. 이를 해결하기 위한 C3 서비스의 선행연구를 TABLE Ⅱ로 정리하였다.

Li[6]는 C3에 대한 최초의 형식적인 설명과 보안 운영 요구사항과 OPRF (Oblivious Pseudo Random Function)[26] 기반의 C3 프로토콜 식별자 기반 버킷화(IDB)를 제시했다. 이는 아이디-패스워드의 해시 접두사 대신 사용자 아이디 해시값의 처음 16bit의 Prefix 값만 버킷 ID로 사용하도록 하여 Hash-Prefix 값에 패스워드 정보가 없으므로 공격자가 온전한 계정-패스워드 쌍의 정보를 추론하기 어렵게 한다. 이는 안전성은 높일 수 있으나 단일의 버킷에 더 많은 데이터를 담게 되는 대형 버킷은 대량의 메시지를 교환하기 때문에 높은 지연시간과 대역폭이 필요하다.

Yu[25]는 기존 C3에서의 사용자 hash-prefix 처리에 대해 묵시적으로 신뢰해야 하는 점을 해결한다. 서버에 쿼리된 계정을 노출하지 않고 클라이언트의 계정 유출 여부를 확인할 수 있는 효율적인 C3 프로토콜을 제안한다. 아이디-패스워드가 아닌 아이디 해시의 처음 β bit를 사용하여 서버 데이터를 여러 개의 블록으로 나누는 데 사용한다. 이를 통해 민감한 패스워드를 숨겨 연산을 수행한다. 그러나, 서버 데이터의 증가는 버킷 데이터를 대량으로 생성할 수 있고, 연산 과정에서 대량의 메시지를 교환하기 때문에 높은 지연시간과 대역폭을 가진다.

기존 C3에서는 사용자의 인풋 데이터와 서버의 데이터가 완전히 같으면 매칭이 된다. 이러한 검색 방법의 한계를 극복하기 위해 Pal[27]은 'Might I Get Pwned'(MIGP)를 통해 사용자 계정을 자격증명 조작 공격으로부터

보호한다. 개인 유사성 테스트(PST)를 통해 사용자가 침해된 패스워드와 유사하거나 같은 패스워드를 선택하지 않도록 사용자에게 경고한다. 그러나 MIGP 서비스에서 생성하는 공개되지 않은 자격증명을 추론하는 공격에 취약하다.

Li[28]는 동형암호 기반의 PSI 프로토콜을 적용하되 복잡성과 지연시간을 줄이기 위해 OPRF 대신 해시함수를 이용하는 privacy preserving password checkup(PIPA)을 설계하였다. 그러나 쿼리 연산 복잡도가 해시값의 길이에 선형적으로 증가하여 높은 쿼리 대역폭을 가진다.

Gunther[29]는 비대칭 데이터 환경에서 연산 효율을 높이기 위한 hash-prefix 값의 노출 문제를 해결하기 위해 서버가 클라이언트의 쿼리 정보를 학습하지 않으면서 클라이언트가 데이터베이스 항목을 안전하게 조회할 수 있는 PIR(Private Information Retrieval)을 사용한다. PIR은 악의적인 서버가 공모하는 것을 방지하기 위해 2대 이상의 멀티서버를 전제로 하며, 온라인 계산의 일부를 오프라인 단계로 이전시켜 쿼리 종속적 전처리 기능을 통해 안전성과 성능 향상을 가져온다. 그러나 제안 기술은 데이터 조회 수준에서 평가되었으며, 교집합 데이터를 추출하는 완전한 C3 프로토콜의 실행 관점에서 측정되지 않았다.

TABLE Ⅱ. C3 선행연구 분석

ref	기여점	한계점
[6]	• C3에서 발생하는 연산 과정의 공격모델을 증명 • PSI 연산 시 안전성 증대를 위해 식별자 기반의 버킷화를 제안함	• 단일의 버킷에 더 많은 데이터를 담게 됨으로써 더 많은 대역폭을 야기

	• 사용자 쿼리에 패스워드 정보가 없어 공격자가 온전한 계정-패스워드 쌍의 정보를 추론하기 어려움	
[25]	• 사용자 쿼리 hash-prefix 값에 사용자 ID만 기반으로 하여 비밀 정보 없는 안전한 매핑 방법을 사용하여 연산 시간 및 통신 대역폭 감소	• 서버 데이터의 증가는 버킷 데이터를 대량으로 생성함 • 대량의 메시지를 교환하므로 높은 지연시간과 서버 대역폭
[27]	• 사용자 쿼리와 서버 데이터의 유사성을 비교하여 유출 위험성을 경고함	• 서비스에서 생성하는 공개되지 않은 자격증명 추론 공격에 취약함
[28]	• 동형암호 기반 PSI 연산으로 연산 효율을 높이고, 안전성 측면에서 사용자 입력 독립성 증명 • 카드번호 및 기밀정보와 같은 비밀 정보를 포함한 확장 가능성을 제안	• 쿼리 연산 복잡도가 해시값의 길이에 선형적으로 증가하며 높은 쿼리 대역폭을 가짐
[29]	• 사용자 쿼리 hash- prefix 값 노출 문제 해결을 위해 개인정보 검색(PIR)으로 안전한 DB 조회를 가능	• 완전한 C3 프로토콜의 실행 관점에서 측정되지 않음

4. 업데이트 가능한 PSI 선행연구 분석

업데이트 가능한 환경을 고려하는 선행연구를 TABLE Ⅲ으로 정리하였다.

KISS[12]는 사전 계산 단계를 통해 온라인 연산의 부담을 줄여 연산 효율성을 높였다. N 크기의 대규모 집합을 보유한 서버와 N'크기의 소규모 집합을 보유한 클라이언트 사이의 PSI 연산의 기본 단계에서는 데이터 독립적으로 사전연산을 수행하고, 설정 단계에서 통신 및 계산 비용이 집합 N에 선형성을 가진다. 마지막 온라인 연산 단계에서 집합 N'에 선형성을 가진다. 본 연구는 클라이언트 입력의 수의 변화에 따라 기본 및 설정 단계를 다시 계산하지 않고 집합을 업데이트하여 클라이언트가 새로운 집합에 대해 온라인 단계를 실행할 수 있다. 그러나 본 연구에서는 온라인 연산 과정에서 클라이언트 원소 추가 및 삭제의 갱신 과정만을 포함하고 있어 서버 데이터의 업데이트 시 연산 효율을 개선할 수 없다.

Abadi[30]는 위임된 다자간의 PSI 환경에서 연산 효율성 문제를 개선한다. 특히, 클라이언트는 암호화된 개인 데이터를 서버에 업로드하여 PSI 계산을 위임하고, 클라이언트의 데이터를 업데이트할 수 있다. 이때, 클라이언트는 업로드 데이터의 크기에 비례해 통신 및 계산 비용을 부담하여 연산비용을 절약한다. 그러나 이는 클라이언트의 원소가 기존 업로드 집합에 추가로 업데이트되므로 서버 데이터에 업데이트 연산 효율성은 고려되지 않았으며, PSI 프로토콜의 계산 및 통신은 전체 집합에 비례하며 이를 부담하여 처리하는 서버가 필요하다.

Badrinarayanan[31]는 업데이트 가능한 PSI를 연구하며, 이는 당사자가 개인 집합의 교집합을 정기적으로 계산할 수 있도록 하면서 또한 집합이 지속해서 추가 및 삭제되는 업데이트 기능을 허용한다. 특히, 데이터 추가 기능 프로토콜은 한 당사자가 암호화된 데이터베이스를 업데이트하고, 다른

당사자는 암호화된 데이터베이스를 탐색하도록 구성한다. 이는 업데이트 가능한 기능을 통해 계산 및 통신 복잡성이 전체 집합이 아닌 업데이트된 집합의 데이터에 비례해 증가한다. 그러나, 이는 Diffie-Hellman 프로토콜 기반으로 PSI 연산을 수행하여 소규모 데이터의 연산 효율성만을 가진다. 대량의 집합에 적용할 경우, 서버 집합을 암호화하여 클라이언트의 저장소에 전송하는 과정이 클라이언트에게 부담이 될 수 있으며, 프로토콜은 전체 집합 크기가 충분히 크거나 새로운 업데이트가 충분히 작을 때 또는 대역폭이 낮은 네트워크에 한정해서 효과성을 가진다.

TABLE Ⅲ. 업데이트 가능한 PSI 분석

ref	기여점	한계점
[12]	• 비대칭 집합에는 연산 비효율을 가지는 기존의 PSI를 분석해 비대칭 PSI 연산에서 효율성 개선 • 서버가 설정 단계를 다시 계산하지 않고 집합 업데이트, 클라이언트가 새로운 집합의 온라인 단계 실행	• 연구에서는 온라인 연산 과정에서 클라이언트 원소 추가 및 삭제의 갱신 과정만을 포함하고 있어 서버 데이터의 업데이트 시 연산 효율을 개선할 수 없는 한계
[30]	• 클라이언트 독립적으로 클라우드에 개인 집합 업로드 후 무제한으로 계산 위임 가능	• 클라이언트 개인 집합을 지속해서 학습하여 안전성을 보장받지 못할 가능성
[31]	• Diffie-Hellman 프로토콜 기반의 업데이트 가능한 PSI 연산을 가능하게 함	• 대량 집합의 연산 효율 저하 • 전체 집합 크기가 충분히 크거나 새로운 업데이트가 충분

• semi-honest adversaries로부터 안전성을 높이고, 업데이트 셋에만 비례한 연산 및 통신 복잡도	히 작을 때 또는 대역폭이 낮은 네트워크에 한정해서 효과성을 가짐

선행연구 분석을 통해 비대칭 데이터의 비교 연산 과정에서 이용자의 정보 일부를 누출하는 문제가 C3에서 발생하고 있으며, 이때 C3 서버가 사용자의 계정정보를 학습하여 추론하는 공격에 대한 대응방안이 없는 문제점이 있다. 또한, 일부 선행연구에서만 C3 특성 중 하나인 데이터의 지속적인 업데이트 기능을 포함한 Diffie-Hellman 기반의 PSI 프로토콜 연산 방안을 제안한다. 그러나, 이는 소량의 데이터 연산에서 효율성을 가지는 프로토콜이며, 다회성 연산을 통해 낭비되는 연산비용이나 다회성 연산 시 효율성을 개선할 수 있는지 일회성 연산을 기준으로 평가하여 효과적으로 증명하고 있지 않다. Diffie-Hellman 기반의 PSI 프로토콜은 대량의 데이터 연산 적용 시 효율성이 열화되는 특징을 가지고 있으므로, 이에 대한 개선이 요구된다.

Ⅲ. 문제점 도출

상기 선행연구 분석을 통해 비대칭 데이터 환경의 PSI 연산 안전성과 효율성 향상 문제, 지속적이고 대량의 크리덴셜 유출에 따른 C3 서버와의 PSI 연산 효율성 보장 문제를 기반으로 본 장에서 세 가지 문제점을 도출한다.

첫째, 기존 PSI 연산에서 비대칭 데이터 환경에서의 PSI 연산으로 발전하면서 대량의 데이터에서 소량의 연관 데이터만을 추출해 비교 연산을 수행한다. 그러나, 소량의 연관 데이터를 추출하는 과정에서 상대 노드에 보유 데이터의 정보를 노출하게 되고, 이를 통해 보유 데이터를 유추하여 밝히고자 하는 악의 행위가 가능하다. 이는 연산 과정에서 사용자의 계정정보를 학습한 공격자가 예측 예산을 대폭 줄이게 되어 C3를 통한 계정 보호 목적을 침해할 수 있으므로 발신 노드에의 수신 노드 정보의 공개를 최소화하여야 한다.

둘째, 유통되는 자격증명의 양이 지속해서 증가하고 있다. 인터넷상에서 150억 개의 도난당한 자격증명이 유통되고 있으며[32], RockYou 2021 유출[33]과 같은 사건에서 대규모의 최신 패스워드 및 7억 명의 LinkedIn 사용자 계정정보가 다크웹에서 유통된다. 또한, 최근 MOAB(Mother of all Breaches) 사건에서 유출 계정의 양이 260억 개에 달한다. ‘21년도 전 세계 88개국에서 샘플링한 79,635건의 침해사고를 분석한 결과, FIGURE 4와 같이 업무 종사 형태에 따라 도난 데이터의 유형에서 자격증명이 60% 이상을 차지하며[34], 실제로 기업에서는 크리덴셜 스터핑으로 인한 고객 신뢰도 하락과 대응을 위한 IT 비용 증가 등으로 연간 평균 600만 달러의 손실을 보고 있다[35]. 서비스 이용자와 제공자가 이러한 피해에 적극적으로 대응하기 위해 C3의 극대량 데이터에서

의 연산 효율성 향상이 필요하다.

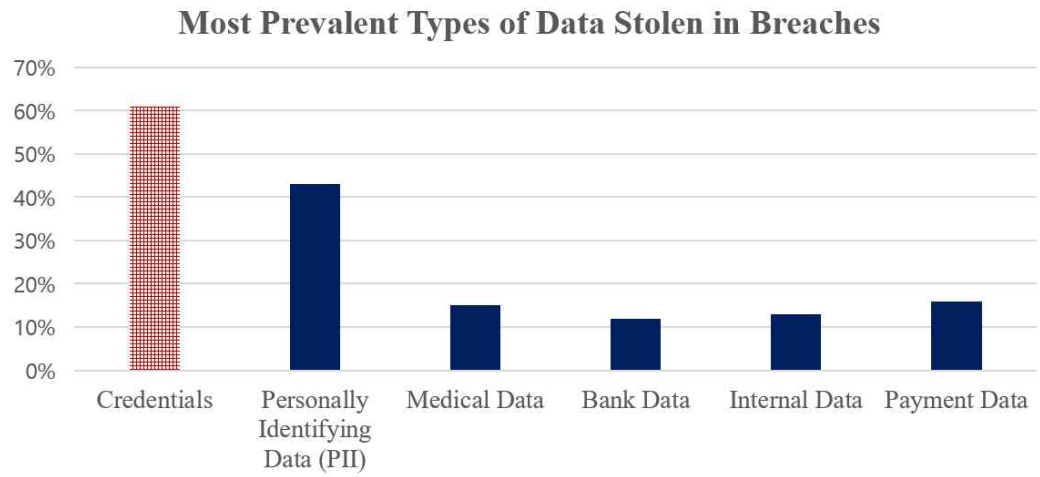

FIGURE 4. 유출 데이터 유형 분류

셋째. 기존의 PSI는 고정된 데이터의 일회성 연산 방안 위주로 연구되어 왔기 때문에 사용자와 서버가 데이터를 업데이트할 경우, 이전 연산 결과를 고려하지 않고 반복된 연산 수행으로 연산 복잡도가 증폭된다. 또한, 서버 측의 다량 데이터가 업데이트되는 환경, 연산비용 절약을 고려하여 C3의 효율성 강화를 위해 연산 복잡도를 상쇄할 방안이 요구된다.

이를 해결하기 위해 다음의 요구사항을 도출하였다. 첫째, 안전한 데이터 난독화 방안이 요구된다. 둘째, 연산 과정의 사용자 쿼리에 동형암호를 적용하여 연산비용을 줄이고 연산 효율을 높인다. 셋째, C3에서 다루는 데이터의 양이 커질 것으로 예측되므로 가입 계정과 유출 계정을 업데이트하는 상황을 고려하여 중복 연산을 방지하는 다회성 PSI 프로토콜이 필요하다.

Ⅳ. Large Asymmetric PSI 연산기법

본 장에서는 수신 노드와 발신 노드가 대량의 비대칭한 데이터와 스토리지, 연산 능력을 보유한 C3 환경에서 사용자 계정의 유출여부를 빠르게 탐지하기 위한 업데이트 가능한 PSI 연산 프로토콜을 제안한다. 기존의 업데이트 가능한 연산은 특히, Diffie-Hellman 알고리즘 기반의 연산을 이용하므로 다음과 같은 한계를 가진다. 첫째, 하루 동안 추가되는 데이터를 기반으로 업데이트 연산을 보인다. 이는 실시간 탐지가 필요한 연산 환경에서는 적용하기 한계를 가진다. 둘째, Diffie-Hellman 알고리즘 기반의 연산은 소량의 데이터 연산에서 효율성을 가지나, 대량의 데이터 연산 환경에서는 연산 시간 및 통신량의 지연으로 인해 효율성이 저하된다. 셋째, 수신 노드와 발신 노드 보유 데이터를 모두 업데이트하는 환경을 가정하여 연산을 수행한다. 해당 연산에서는 기존 데이터양이 많고, 업데이트 데이터양이 적을 때만 효율을 가진다. 이러한 한계점을 개선하고, C3에서 가능한 공격 시나리오에 대한 방어 가능성을 밝힌다. 특히, C3 서비스의 주요 보안 요구사항은 사용자 계정의 비밀성으로, 본 연구에서는 semi-honest 발신 노드가 수신 노드의 쿼리 계정을 학습하여 계정 유추 공격 확률을 높이는 공격과 malicious 수신 노드가 발신 노드가 보유한 다량의 타인 계정정보에 대한 학습 공격에 대응할 수 있다. 본 연구에서는 이러한 위협모델에 강인한 Large Asymmetric PSI 연산기법을 제안한다.

1. 구조 및 동작 원리

Large Asymmetric PSI 연산기법은 다음의 특징을 가진다. 첫째, 데이터를

기반으로 tag를 생성하여 업데이트되는 시간 간격의 영향을 받지 않고, 데이터의 업데이트 여부에 따른 실시간 연산, 다회성 연산을 가능하게 한다. 둘째, 효율적인 데이터 난독화 과정, 데이터 교환 연산을 수행하기 위해 대량의 데이터를 가진 발신 노드 측에서 Oblivious pseudo random function(OPRF) 연산을 수행하고 동형암호를 이용한 대량의 데이터의 효율적인 연산을 가능하게 한다. 셋째, C3의 특성을 반영하여 수신 노드의 데이터는 계정을 변경할 때만 업데이트되고, 발신 노드의 데이터는 서비스 제공자 측에서 대량의 유출 계정을 신속히 업데이트하여 연산 효율성을 향상시킨다. 이 과정에서 OPRF 연산은 수신 노드의 데이터를 발신 노드로부터 보호하고, 연산 과정에서 생성된 tag와 랜덤 값 r을 통해 발신 노드의 데이터를 수신 노드로부터 보호한다.

FIGURE 5는 제안하는 Large Asymmetric PSI 시스템의 간단한 구조를 보여준다.

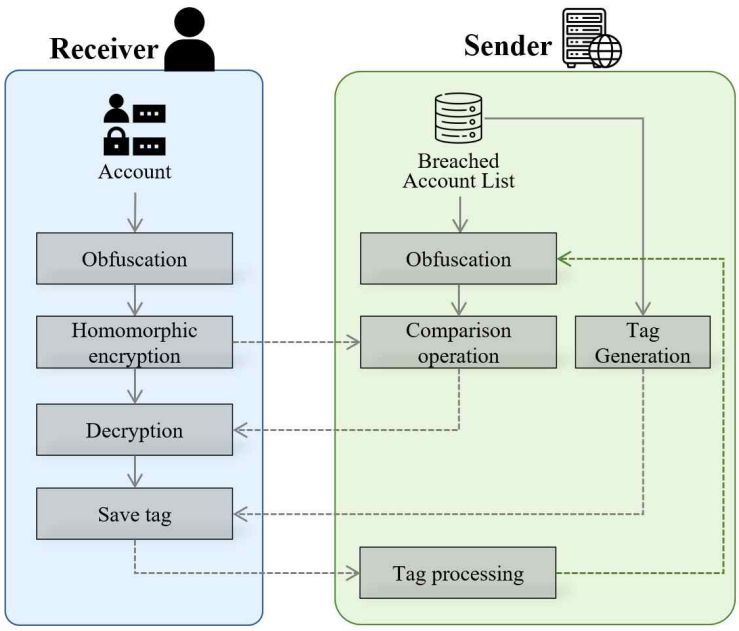

FIGURE 5. Large Asymmetric PSI 아키텍처

발신 노드는 대량의 보유 데이터를 전처리하여 연산 시간을 절약한다. 전처리 시 발신 노드의 보유키로 보유 데이터 난독화 연산을 수행하고, 수신 노드의 인풋 데이터를 같은 키로 난독화 하여 수신 노드에 전달한다. 수신 노드는 수신 값을 동형 암호화하여 비교 연산을 요청한다. 발신 노드는 암호화 값을 받아 서버 데이터 노출 방지를 위한 안전한 다항식 연산을 수행하고, 해당 연산 결과를 전송한다. 수신 노드는 해당 값을 받아 복호화하여 결괏값을 확인한다. 본 논문에서는 다회 연산을 통해 서버의 연산 데이터양을 줄이기 위해 수렴 암호화 기반의 *tag*를 이용해 다회 연산에 필요한 데이터만을 추출한다. *tag* 기능을 이용하지 않으면 서버의 연산 과정에서 다량의 데이터와 중복된 연산을 수행하므로 연산 시간 및 연산량이 낭비된다. 반면, *tag* 기능을 이용하면 이전 연산에 사용한 발신 노드와의 연산 데이터를 서버가 복구하여 이를 제외하고 연산에 필요한 데이터만을 추출해 연산을 수행할 수 있으므로 연산 복잡도를 줄일 수 있다.

1) 발신 노드 오프라인 전처리 연산

Diffie-Hellman 프로토콜은 데이터양에 연산 복잡도가 비례한다. 따라서, Large Asymmetric PSI 연산기법에서는 OPRF 연산을 통해 데이터를 보호하고, 수신 노드와의 통신 이전에 전처리 과정에서 대량의 데이터에 대한 연산을 수행하여 연산비용을 줄인다. 또한, 동형암호화를 통한 데이터 처리를 위해 연산의 복잡도를 줄여 효율성을 높인다. FIGURE 6은 이를 적용해 오프라인에서 수행하는 발신 노드의 전처리 연산을 정의한다. 발신 노드의 집합 $X=\{x_1,...,x_n\}$와 수신 노드의 집합 $Y=\{y_1,...,y_n\}$이 있을 때, $n^y \ll n^x$이며, 수신 노드만 연산 결과 $X \cap Y$를 알 수 있다. 연산 과정은 오프라인 연산과 온라인 연산 과정을 가진다. 오프라인 연산에서는 대량의 발신 노드의 데이터에 대해 OPRF 및 해싱 연산을 수행해 데이터를 난독화 하여 각자 보유한 데이

터 원본을 노출하지 않아 유추 공격의 한계를 가져온다.

Input :
발신 노드의 집합 $X = \{x_1,...,x_n\}$
수신 노드의 집합 $Y = \{y_1,...,y_n\}$, $n^y \ll n^x$
Output : 수신 노드는 $Result = X \cap Y$; 발신 노드는 $\perp$

1. 발신 노드(서버)
(1) 발신 노드의 OPRF 수행
발신 노드 셋에 대해 샘플링된 보유 키 k를 가지며, OPRF $F: \{0,1\}^* \rightarrow \{0,1\}^k$ 이다. OPRF 연산을 수행하여 $X' = \{H(F_k(x) : x \in X\}$를 도출한다. 이때 H는 랜덤 오라클 해시함수이다.

(2) 해싱
파라미터 m은 Cuckoo Hashing으로 N_y의 공을 m개의 바구니 매핑에 이용한다. 예를 들어 랜덤한 해시함수 3개 $h_1, h_2, h_3 : [0.1]^\sigma \rightarrow [m]$이고, 모든 $x \in X'$를 $B[h_1(x), h_2(x), h_3(x)]$에 삽입한다. 발신 노드의 보유 데이터는 오프라인의 전처리 과정을 통해 연산비용을 절약한다.

(3) 파라미터 선정
발신 노드는 수신 노드와 공유할 연산에 필요한 파라미터 값을 선정한다. 또한, 동형암호 연산 과정에서 필요한 연산 깊이 수를 선정한다.

FIGURE 6. Large Asymmetric PSI 전처리 연산

2) 수신 노드와 발신 노드의 통신 및 연산

수신 노드와의 통신을 통해 발신 노드는 파라미터를 전송하고, 수신 노드의 입력 데이터를 받아 OPRF 연산을 수행한다. 이후 해시함수를 이용해 랜덤하게 난독화하고 동형암호화하여 발신 노드에 전송한다. 발신 노드는 $Enc(w) = r*(Enc(x_n) - y_n)*(Enc(x_n) - y_{n+1})*(Enc(x_n) - y_{n+2}) +,.,+ Enc(x_n)$로 수

신 노드 데이터와의 교집합 연산을 수행한다. 이때, 랜덤 값 r을 통해 복호화 후 발신 노드의 데이터가 노출되어도 r값을 통해 보호될 수 있도록 한다. 연산 이후 결괏값 $Enc(w)$를 전달한다. 수신 노드는 이를 복호화하여 $Enc(w)$가 X의 원소일 경우 $y \in X$이며, $Enc(w)$가 랜덤한 값일 경우 $y \notin X$이다.

2. 수신 노드(클라이언트)
(1) 수신 노드의 OPRF 수행
수신 노드 데이터의 OPRF 연산 수행을 요청한다. 발신 노드는 키 k로 수신 노드의 데이터 Y를 연산해 $F_k(y)$를 생성하고, $Y' = \{H(F_k(y) : y \in Y\}$를 보유한다.
(2) Cuckoo Hashing
수신 노드는 Cuckoo Hashing을 이용해 보유 데이터 Y'를 해시함수를 가지는 h_1, h_2, h_3를 사용하는 m개의 bins를 보유한 table C에 매핑한다.
(3) 동형암호화
수신 노드는 보유 데이터에 대해 완전 동형암호화하여 발신 노드에 보낸다.

3. 발신 노드(서버)
(1) 동형암호 연산
발신 노드는 수신 노드의 데이터와 발신 노드의 데이터를 교집합 연산하여 $Enc(w) = r*(Enc(x_n) - y_n)*(Enc(x_n) - y_{n+1})*(Enc(x_n) - y_{n+2}) + ,,, + Enc(x_n)$를 도출한다.

4. 수신 노드(클라이언트)
(1) 복호화 및 결과 확인
결과적으로 $Enc(w)$가 X의 원소값일 경우 $y \in X$이며, 랜덤한 값일 경우 그렇지 않은 것으로 확인한다.

FIGURE 7. Large Asymmetric PSI 통신 및 연산 과정

3) 수신 노드 및 발신 노드 데이터 업데이트 연산

기본 연산 과정은 다음과 같다. 최초 연산에서 발신 노드는 대량의 데이터

에 대해 전처리 연산을 수행하고, 수신 노드의 입력값을 받아 같은 처리를 수행한다. 이후, 수신 노드는 입력값을 암호화하여 전송하고, 발신 노드는 보유 데이터와 수신 데이터와의 연산 결괏값을 전송한다. 수신 노드는 해당 값을 복호화하여 결과를 확인한다.

Algorithm 1 Tag Transmission Protocol

procedure SENDER($X' = X_{old} + X_{new}, tag, Enc(w)$)
  $key \leftarrow H(X_{old})$
  $ct \leftarrow Enc(key, X_{old})$
  $tag_{Y \cap X_{old}} \leftarrow H(ct)$
  Add $(tag_{Y \cap X_{old}} : (key, ct))$ hash table
  Send $(tag_{Y \cap X_{old}}, Enc(w))$ to receiver
end procedure
procedure RECEIVER($Y, tag, Enc(w)$)
  if $Enc(w) \ni Y$ **then**
    $tag \leftarrow$ Null
  else
    $tag \leftarrow tag_{Y \cap X_{old}}$
    Send tag to sender
  end if
end procedure
procedure SENDERFINALIZE(tag)
  Retrieve (k, ct) from sender's hash table using tag
  $X_{old} \leftarrow Dec(k, ct)$
  $X_{update} \leftarrow X' - X_{old}$
end procedure

FIGURE 8. tag Processing pseudo-code

FIGURE 8은 수신 노드와 발신 노드의 데이터 업데이트 가능한 PSI 매커니즘이다. tag값은 발신 노드와 수신 노드 간의 연산에 이용한 발신 노드의 보유 데이터 X로 생성되며, 메시지 기반 암호화 방식으로 구성한다. X에 대해 대칭키 해시함수를 통해 암호화 키를 생성하고, 생성한 키로 X를 암호화

한다. 도출된 암호문 ct를 해시함수로 tag값을 생성한다. 이때 서버는 해시 테이블에 tag값에 ct값, 암호 키를 매칭해 저장한다. tag와 결괏값을 함께 수신한 사용자는 보유한 키를 이용해 복호화하여 결괏값을 확인하고, tag의 카운터를 증가시켜 수신한 tag를 업데이트하고 저장한다. 이때, tag 값에는 데이터의 원본 값을 포함하고 있지 않아 발신 노드 보유 데이터를 유추할 수 없다. 수신 노드는 계정 유출을 확인한 경우 계정을 변경하면 tag 값을 초기화한다. 유출 계정이 없는 것으로 확인되면 추후 연산에서 tag를 이용해 연산을 수행한다. 수신 노드는 이후 다회 연산을 수행할 때 발신 노드에 tag 값을 전송한다. 이를 수신한 수신 노드는 tag를 저장한 해시 테이블과 매칭하여 원문을 복구하고, 업데이트된 DB 셋에서 이전 연산을 수행한 X_{old} 값을 제외한 X_{update}만을 추출하고 이후 연산 과정을 거친다.

2. 전송 매커니즘

FIGURE 9는 Large Asymmetric PSI의 전체 연산 매커니즘이다. 수신 노드와 발신 노드는 입력 데이터 X와 Y를 선정하고, 발신 노드는 대량의 데이터의 연산 부담을 줄이기 위해 전처리로 난독화 연산을 수행한다. 이후 Y를 수신하여 난독화 연산을 수행한 후 전송하고, 수신 노드로부터 암호화된 값을 받아 비교 연산을 수행한다. 이때, 연산 결괏값 $Enc(w)$와 연산 데이터 X의 위칫값을 기반으로 생성된 tag값을 전송한다. 수신 노드는 $Enc(w)$를 복호화하여 결과를 확인할 때, 유출일 경우 계정의 패스워드를 변환한다. 이때는 Y값이 변경되었으므로, 수신한 tag값을 초기화하여 추후 연산에서 사용하지 않도록 한다. 반면 수신 노드가 유출 계정이 아닌 것으로 확인한 경우는 유출임을 확인할 때까지 재연산을 요청할 수 있다.

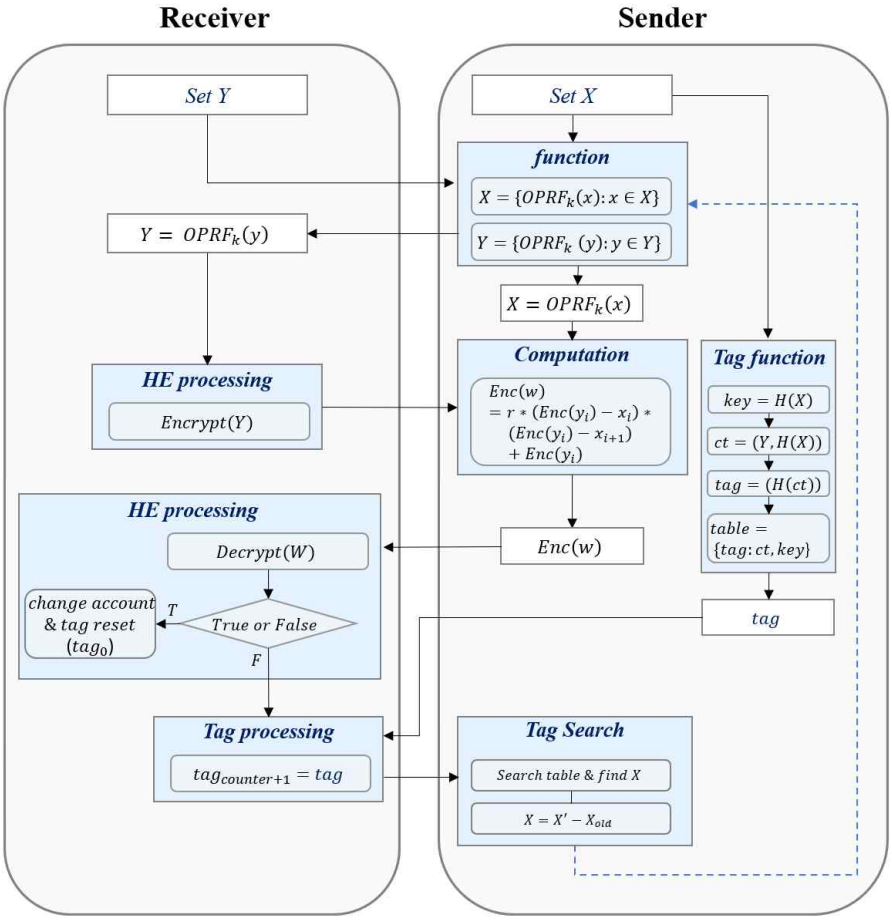

FIGURE 9. Large Asymmetric PSI 연산 매커니즘

추후 연산 요청 시 발신 노드에서는 수신 tag값을 이용해 재연산임을 확인하고, tag를 복구하여 새롭게 추가된 전체 집합에서 이전의 연산한 데이터의 위칫값을 제외하는 $X' - X_{old}$ 연산의 결과로 새롭게 추가된 X_{update}만을 추출하고 전처리 연산을 수행한다. 다회성 연산의 tag값은 추가 데이터를 포함해 X'로 tag를 생성하는 과정을 반복한다. 이를 통해 낭비되는 연산비용을 줄여 연산 효율을 높인다.

이러한 연산은 C3의 주요 보안 요구사항인 사용자 계정의 비밀성을 유지할 수 있다. OPRF를 통한 수신 노드 데이터의 난독화 과정에서 발신 노드에게 완전히 노출하지 않고, 발신 노드는 이를 유추할 수 없다. 발신 노드가 보내는 *tag* 값은 보유 데이터의 원본을 포함하고 있지 않아 공격자가 발신 노드 보유 데이터를 유추할 수 없다. 또한 동형연산 결괏값에 대해 랜덤 값 r을 연산은 과정을 통해 결괏값을 추가로 보호함으로써 공격자가 수신 노드가 발신 노드가 보유한 다량의 타인 계정정보에 대한 학습 공격에 대응할 수 있다.

Ⅴ. 실험 환경 및 성능 평가

1. 실험 환경

1) 실험 방법 및 환경

종래 방식인 Static PSI와 제안하는 Large Asymmetric PSI의 연산 속도 및 효율을 비교 평가하기 위해 TABLE Ⅳ을 이용해 실험 환경을 구축하였다. 최초 연산(Initial Computation)과 다회성 연산(Repeated Computation)을 테스트하고자 Ubuntu 20.04.6 LTS 환경에 C++ 기반의 Asymmetric PSI 라이브러리[36]를 이용해 수신 노드 및 발신 노드가 보유한 데이터양에 따라 소요되는 연산 시간과 연산량, 통신량 등의 연산비용을 측정하였다. 또한, 다회성 연산에 이용되는 *tag*를 생성하고 처리하는 과정을 Python 3.0으로 보유 데이터 크기에 따라 time 모듈을 이용해 처리 시간을 시뮬레이션하였다.

TABLE Ⅳ. 하드웨어 실험 환경

실험단계	전체 연산	tag 처리 연산
OS	Ubuntu 20.04.6 LTS	Windows 10
CPU	12th Gen Intel(R) Core(TM) i9-12900	Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
RAM	32.0GB	32.0GB
DISK	5TB	500GB

2) 파라미터

본 연구에서는 발신 노드가 극대량의 데이터를 보유하고, 서버의 데이터 업데이트양을 대량으로 보유할 때, 수신 노드는 소량의 데이터를 가지는 상황을

가정하여 파라미터를 설정하고, 다회성 연산 시의 연산 효율을 증명한다. 발신 노드의 보유 데이터양을 2^{20}, 2^{22}, 2^{24}으로, 수신 노드의 보유 데이터양을 2^{10}로 설정하였으며, 발신 노드의 업데이트 데이터를 2^{14}, 2^{16}, 2^{18}으로 고정하여 TABLE Ⅴ와 같이 다회 연산을 수행하였다.

TABLE Ⅴ. 실험 데이터

데이터 환경	$\|X\|$	$\|X_{update}\|$	$\|Y\|$
Setting 1	2^{20}	2^{14}	2^{10}
Setting 2	2^{22}	2^{16}	2^{10}
Setting 3	2^{24}	2^{18}	2^{10}

3) 평가지표

데이터 보유량에 따른 Large Asymmetric PSI와 Static PSI의 처리 성능을 비교하기 위해 다음의 지표를 이용한다. 첫 번째, 발신 노드와 수신 노드 간의 전체 통신량을 비교한다. 두 번째, 수신 노드의 전송량을 비교한다. 세 번째, 발신 노드의 전송량을 비교한다. 네 번째, 발신 노드의 전처리 연산 시간을 비교한다. 다섯 번째, 를 생성하고 tag값을 수신하여 연산 원본 데이터를 복구하는 과정에 걸리는 시간을 데이터양에 따라 비교한다.

FIGURE 10의 최초 연산(Initial Computation)은 최초 설정한 파라미터에 따라 테스트를 수행한다. 수신 노드는 2^{10}의 데이터를 생성하여 보유하고 있고, 발신 노드가 2^{20}, 2^{22}, 2^{24}을 가지는 세 가지의 비대칭 데이터 환경의 연산을 테스트한다. 본 연산에서는 종래기술과 Large Asymmetric PSI 모두 같은 연산 부담을 가진다. 이에 대한 전체 통신량 및 수신 노드와 발신 노드 각

각의 전송량과 전처리 연산 시간을 측정한다. 이때 발신 노드는 연산 데이터를 바탕으로 *tag*를 생성하고, *tag*의 원본 복구에 필요한 연산 과정 데이터를 테이블로 저장하는 연산비용을 측정한다. FIGURE 11의 다회성 연산(Repeated Computation)의 연산에서는 수신 노드가 이전 연산에서 교집합 원소를 발견하지 못한, 계정정보가 유출되지 않았음을 확인한 것을 전제로 기존에 보유한 2^{10}의 데이터를 다시 불러오고, 최초 연산에서 저장한 *tag* 값을 발신 노드에 전송한다. 발신 노드는 기존의 세 가지 비대칭 환경인 2^{20}, 2^{22}, 2^{24}의 데이터를 보유하되, 각각 2^{14}, 2^{16}, 2^{18} 만큼의 데이터를 추가하여 다회 연산을 수행한다. 이때 발신 노드에서 *tag*를 통해 업데이트 데이터를 추출하는 연산비용을 측정한다.

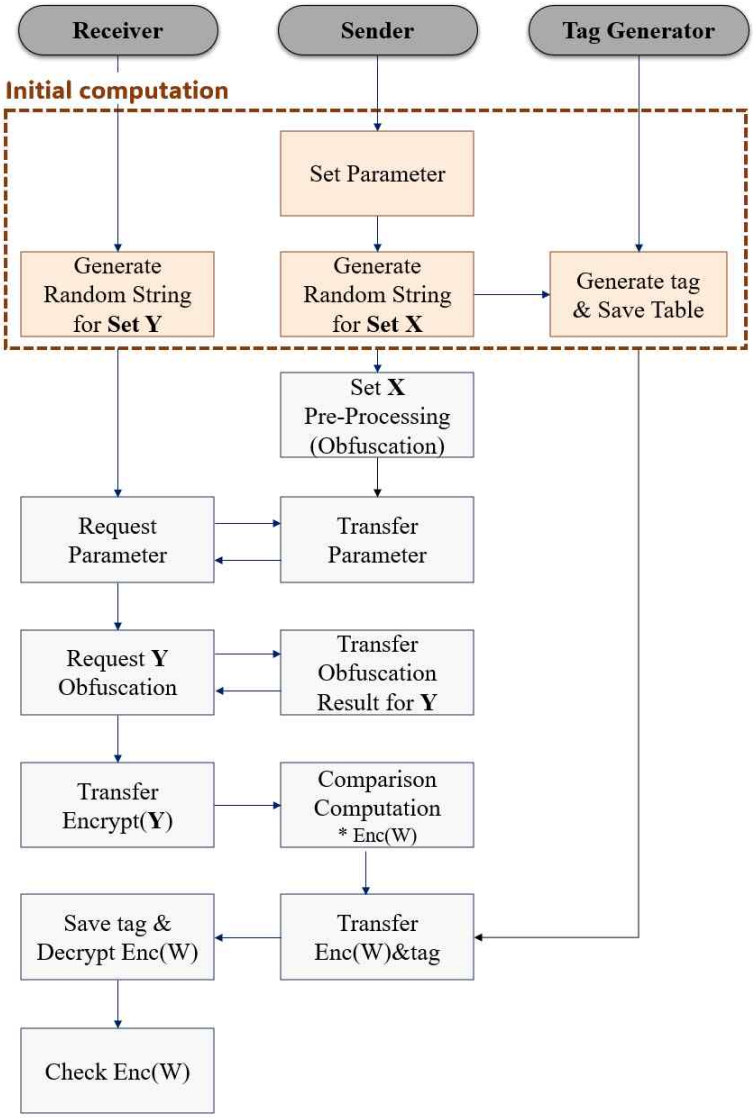

FIGURE 10. Large Asymmetric PSI 최초 연산

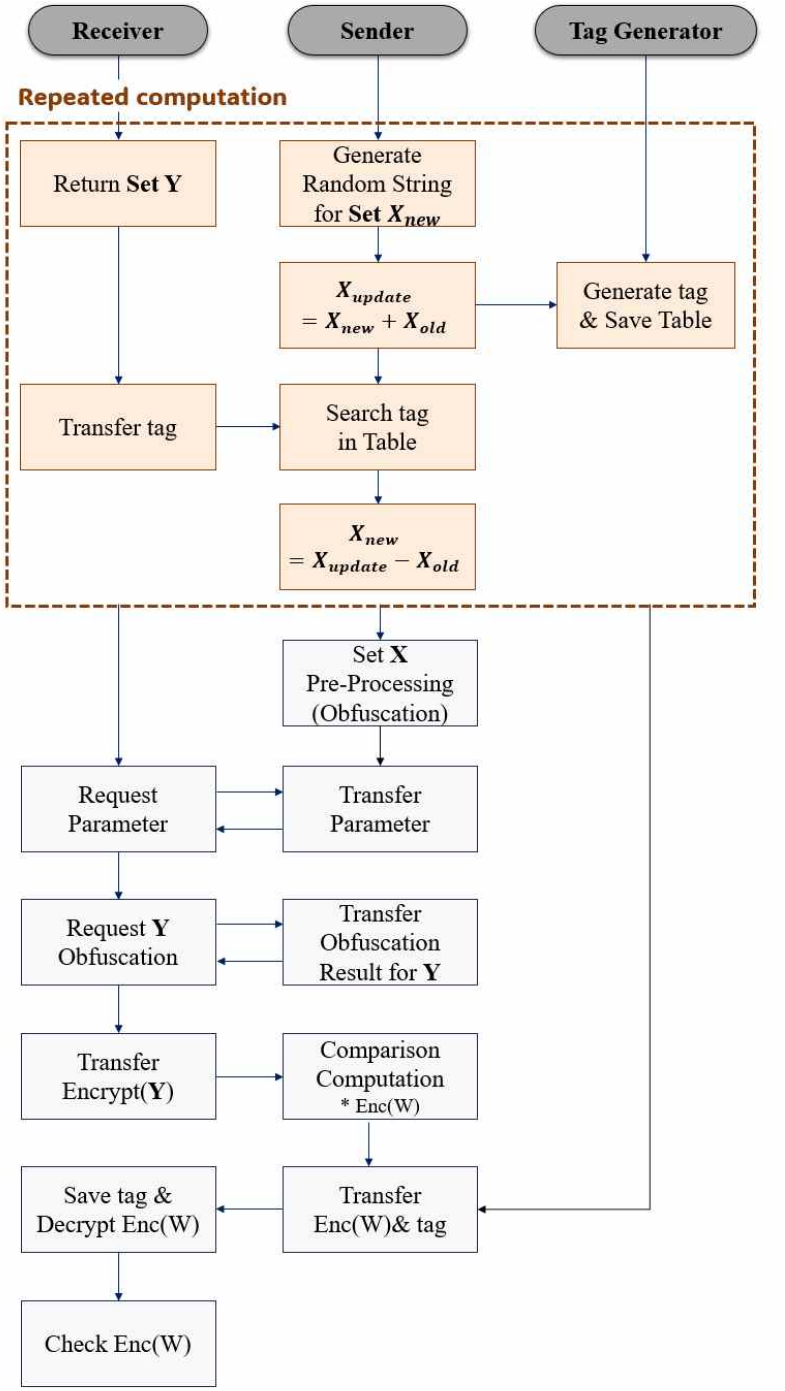

FIGURE 11. Large Asymmetric PSI 다회성 연산

2. 성능 평가

1) 전체 통신량 비교

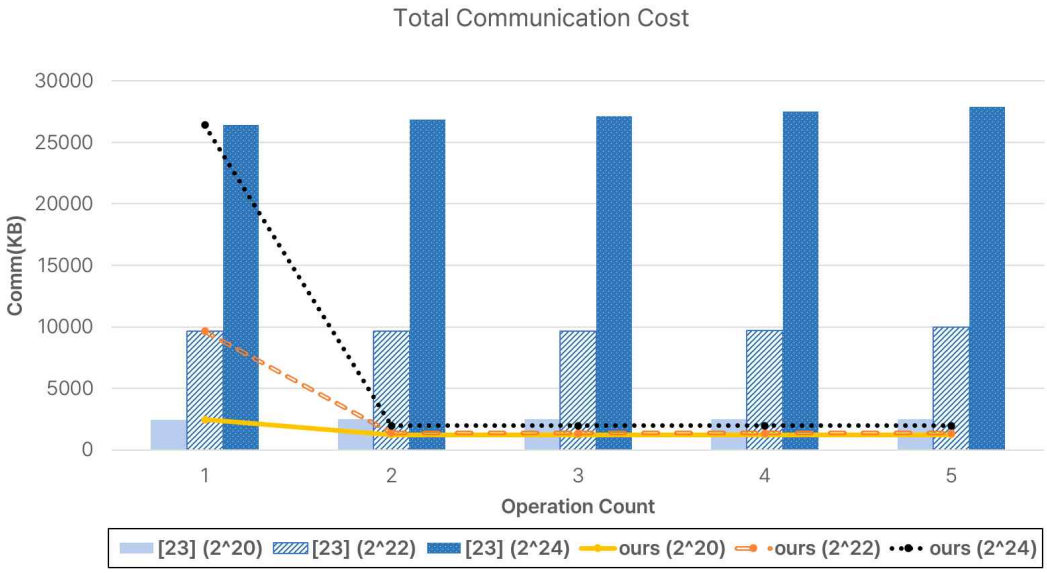

FIGURE 12. 전체 통신량 비교

FIGURE 12와 TABLE Ⅵ는 파라미터별 Static PSI와 Large Asymmetric PSI의 전체 연산량을 보여준다. 발신 노드 측인 서버가 보유한 데이터의 양을 연산 횟수의 증가와 함께 일정하게 증가하도록 설정함에 따라 Static PSI의 통신량이 증가한다. 보유 데이터양을 보존하고 업데이트 데이터가 추가되는 다회 연산을 수행하기 때문에 추가연산을 수행함에 따라 연산 효율의 저하를 가져온다. 반면, Large Asymmetric PSI는 최초 보유 데이터 기반의 연산을 수행한 후 tag를 기반으로 중복 연산을 방지하며 일정한 업데이트 데이터의 크기에 따른 일정한 통신량을 가지며 Static PSI의 연산보다 효율을 가진다. 특히, $X=2^{20}$에서 2차 연산부터 x2.0배의 효율을, $X=2^{22}$에서 x7.0~x7.3배의 효율을, $X=2^{24}$에서 경우에는 x13.7~x14.2 효율을 가진다. 이처럼 기존의 보유 데이터양이 방대하고 업데이트 데이터양이 방대할 경우

더 큰 효율을 보임을 알 수 있다. 이는 PSI 연산은 보유 데이터양에 따라 통신 복잡도가 증가하기 때문에 tag를 통해 통신 비용을 절약할 수 있음을 알 수 있다.

TABLE Ⅵ. 전체 통신량 비교

X	Y	protocol	연산 횟수				
			1	2	3	4	5
2^{20}	2^{10}	[23]	2452	2485	2486	2519	2519
		ours	2452	1235	1233	1235	1234
2^{22}	2^{10}	[23]	9642	9641	9644	9726	9979
		ours	9642	1366	1363	1362	1366
2^{24}	2^{10}	[23]	26402	26874	27100	27522	27879
		ours	26402	1955	1955	1957	1954

2) 수신 노드의 전송량 비교

FIGURE 13과 TABLE Ⅶ는 파라미터별 Static PSI와 Large Asymmetric PSI의 연산 과정에서 수신 노드가 발신 노드에 전송하는 통신량을 보여준다. 수신 노드 측인 클라이언트의 쿼리 크기를 $Y=1024$로 일정하게 유지함에 따라 Static PSI에서 수신 노드의 전송량은 다회 연산에서도 유사하게 유지됨을 확인할 수 있다. 그러나, 대량의 데이터를 보유한 발신 노드와의 통신은 추가적인 연산비용을 소모하게 한다.

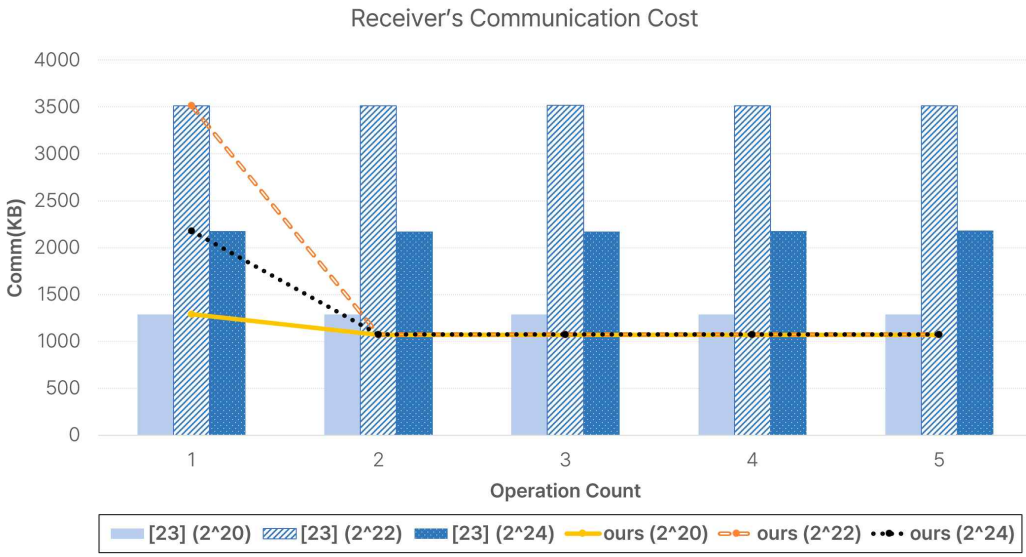

FIGURE 13. 수신 노드의 전송량 비교

Large Asymmetric PSI는 tag를 통해 발신 노드 데이터의 중복 연산을 방지하고, 이전 연산 데이터를 제외한 추가된 데이터와 비교 연산을 수행하기 때문에 수신 노드의 통신량이 감소한다. 이에 따라 Static PSI는 데이터양에 따른 통신 복잡도의 비용이 소모되나, Large Asymmetric PSI는 더욱 낮은 비용으로 연산의 효율성을 가져온다. 특히, 발신 노드의 보유 데이터양과 업데이트 데이터양의 차이에도 불구하고 유사한 통신 복잡도를 가지며, 특히, $X=2^{20}$에서 2차 연산부터 x1.2배의 효율을, $X=2^{22}$에서 x3.2배의 효율을, $X=2^{24}$에서 경우에는 x2.0배의 효율을 가진다. 결과적으로 수신 노드 측의 보유한 데이터의 양을 고정함에 따라 발신 노드의 데이터가 소량일 경우, Static PSI와 Large Asymmetric PSI는 미세한 효율을 보이나, 발신 노드의 데이터가 대량일 경우 Large Asymmetric PSI를 통해 통신 비용을 x2~x3.2배 절약할 수 있음을 알 수 있다.

TABLE Ⅶ. 수신 노드의 전송량 비교

X	Y	protocol	연산 횟수				
			1	2	3	4	5
2^{20}	2^{10}	[23]	1290	1289	1290	1290	1290
		ours	1290	1071	1070	1072	1071
2^{22}	2^{10}	[23]	3515	3514	3518	3515	3514
		ours	3515	1073	1073	1071	1073
2^{24}	2^{10}	[23]	2179	2171	2173	2178	2181
		ours	2179	1074	1074	1075	1073

3) 발신 노드의 전송량 비교

FIGURE 14와 TABLE Ⅷ은 파라미터별 Static PSI와 Large Asymmetric PSI의 연산 과정에서 발신 노드가 수신 노드로의 전송량을 보여준다. Static PSI는 발신 노드 측의 데이터의 보유량에 따라 전송량이 영향을 받음을 알 수 있다.

Large Asymmetric PSI는 *tag*를 통해 발신 노드 데이터의 중복 연산을 방지하고, 이전 연산 데이터를 제외한 추가된 데이터만을 다뤄 통신하므로 Static PSI 대비 다회성 연산에서는 통신량이 크게 감소한 것을 확인할 수 있다. 특히, $X=2^{20}$에서 x7.3~7.5배의 효율을, $X=2^{22}$인 경우에는 x20.9~22.0배의 효율을, 2^{24}인 경우에는 x28.0~29.1배의 효율을 가진다. 결과적으로 데이터 처리량 자체가 감소함에 따라 통신 효율성의 증가를 확인할 수 있다.

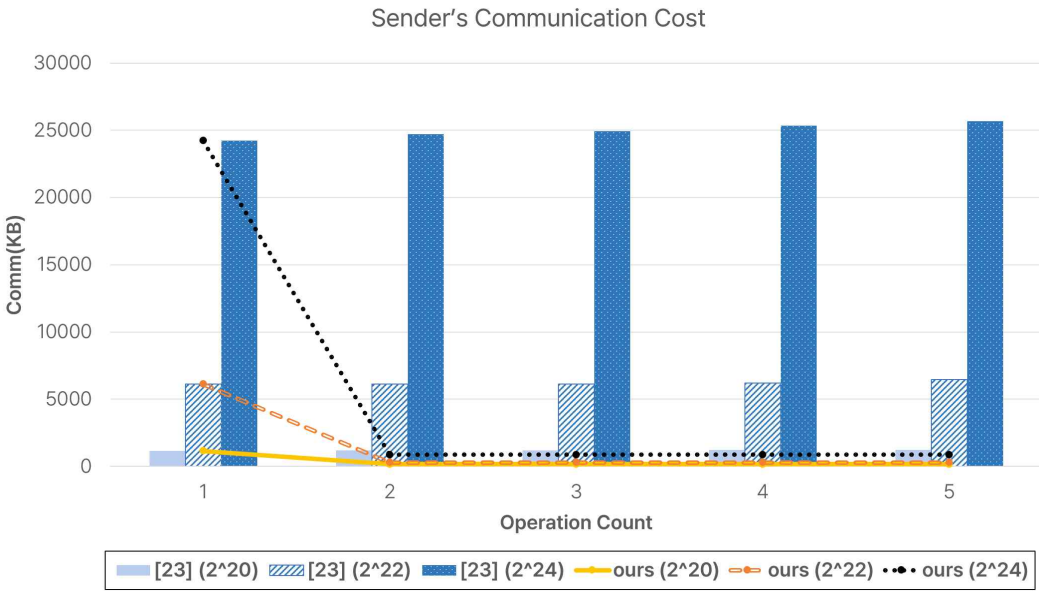

FIGURE 14. 발신 노드의 전송량 비교

TABLE Ⅷ. 발신 노드의 전송량 비교

X	Y	protocol	연산 횟수				
			1	2	3	4	5
2^{20}	2^{10}	[23]	1162	1196	1196	1229	1229
		ours	1162	164	163	163	163
2^{22}	2^{10}	[23]	6127	6127	6126	6211	6465
		ours	6127	293	290	291	293
2^{24}	2^{10}	[23]	24223	24703	24927	25344	25698
		ours	24223	881	881	882	881

4) 전처리 연산 시간

발신 노드는 대량의 데이터를 보유하며, 이에 드는 연산 복잡도를 완화하기 위해 데이터 난독화 과정 일부를 오프라인에서 수행한다. 이를 발신 노드의 전처리 연산이라 한다. 선정한 파라미터 값에 따른 발신 노드의 전처리 시간은 TABLE Ⅸ에 나타내었다. 결과적으로 데이터 처리량 자체가 감소함에 따라 Large Asymmetric PSI의 연산 시간 효율이 대폭 증가하였다.

TABLE Ⅸ. 발신 노드의 전처리 연산 시간 비교

X	Y	protocol	연산 횟수				
			1	2	3	4	5
2^{20}	2^{10}	[23]	00:03	00:03	00:04	00:04	00:04
		ours	00:03	00:00	00:00	00:00	00:00
2^{22}	2^{10}	[23]	00:13	00:15	00:15	00:15	00:15
		ours	00:13	00:01	00:01	00:01	00:01
2^{24}	2^{10}	[23]	01:08	01:10	01:11	01:12	01:14
		ours	01:08	00:01	00:01	00:01	00:01

5) *tag* 연산 소요시간

tag 생성 및 처리 과정을 시뮬레이션하기 위해 32bit의 문자열 데이터를 리스트 형태로 생성하고, 처리하는 과정을 Python을 통해 구현하였다. 서버 보유 데이터의 크기를 10^3, 10^4, 10^5, 10^6, 10^7로 지정하여 데이터 생성 시간, *tag* 생성 시간, *tag* 복구 시간을 측정한 결과를 TABLE Ⅹ로 나타내었다.

TABLE X. *tag* 연산 소요시간

X_{old}	X_{new}	데이터 생성 시간	*tag* 생성 시간	*tag* 복구 시간
10^3		0.02	0.04	0.0
10^4		0.24	0.05	0.0
10^5	$X_{old}*0.5$	2.32	0.05	0.0
10^6		23.85	0.04	0.01
10^7		236.31	0.05	0.10

생성 데이터양에 가장 큰 영향을 받는 것은 데이터 생성 시간이다. 그러나 PSI 연산에서는 데이터를 이미 보유하고 있는 상황으로 고려대상이 아니다. FIGURE 15은 *tag*생성 시간과 *tag* 복구 시간을 누적형 그래프로 나타내 데이터양의 증가에 따라 복구 시간이 증가하고 있음을 확인할 수 있다.

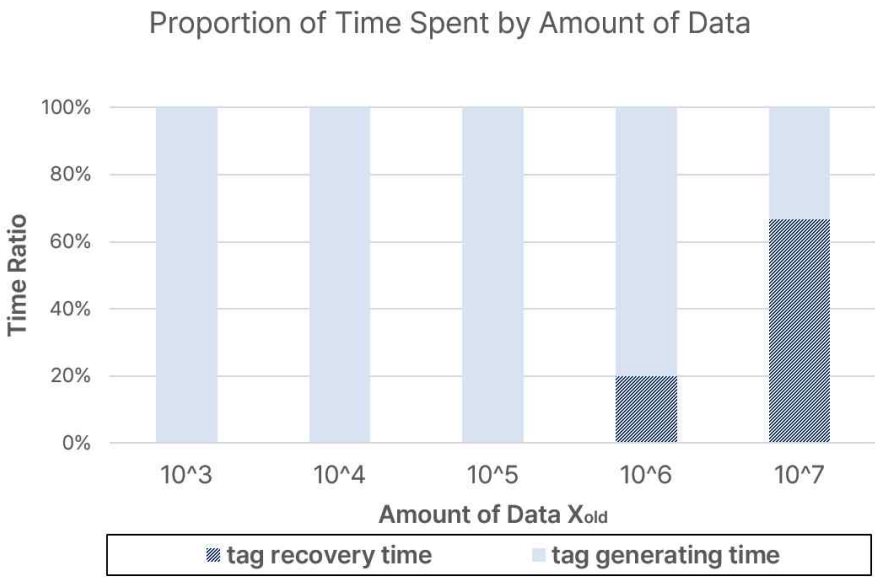

FIGURE 15. 데이터양 별 소요시간 비율

tag생성 시간은 암호 키 key와 암호문 ct를 생성하는 과정을 포함하므로 데이터양에 제한되지 않고 유사한 시간이 소요된다. tag 복구 시간은 tag를 수신하여 매핑 테이블에서 ct와 key를 탐색하는 과정과 업데이트된 X에서 X_{old}를 제외하는 연산 시간을 의미하며, 데이터의 크기에 따라 데이터를 탐색하고 복구하는 과정에 연산 시간을 소요한다. 결과적으로 Large Asymmetric PSI 연산 시 tag로 인한 시간은 비용을 상쇄하며 연산 효율성을 보이며, 특히 X의 양이 대량일수록 Large Asymmetric PSI의 연산 효율을 더 크게 가진다.

Ⅵ. 결론

국내외 개인정보 유출 사건 빈도와 유출 데이터양 증가에 따라 유출 계정정보가 다크웹을 통해 유통되고, 이를 활용한 크리덴셜 스터핑 공격이 침해사고의 원인으로 작용하고 있다. 이에 대응하기 위해 C3는 사용자의 데이터를 보호하여야 하는 서비스 제공자와 자신의 정보를 보호하고 싶은 서비스 이용자 측면에서 모두에게 이점을 가진다. 그러나, 정적인 데이터 환경에서 PSI 프로토콜을 활용한 C3는 유출 데이터 증가와 반복 요청으로 연산 복잡도를 대폭 상승시킬 수 있으며, 이를 상쇄할 수 있는 업데이트 가능한 PSI 연산 방안 연구는 거의 이루어지지 않고 있다. 본 연구에서는 동적인 데이터 환경에서의 효율적인 PSI 연산 방안을 제안하였다. 데이터양에 따른 연산비용과 통신량을 비교하기 위해 동적인 환경을 지원하고 프로토콜의 연산 효율성 향상을 위한 tag값 기반의 업데이트 연산과 정적연산을 비교하였다. 발신 노드 측이 데이터 X의 크기를 대량 보유하고, 수신 노드 측이 데이터 Y를 소량 보유한 비대칭 환경에서 $X=2^{20},2^{22},2^{24}$, $Y=2^{10}$로, 업데이트 데이터양을 각각 $X_{update}=2^{14},2^{16},2^{18}$ 설정하여 5회 업데이트 연산을 수행하였을 때의 통신량, 전처리 연산 시간, tag 연산 시간을 비교하였다. 결과적으로 X의 양이 대량일수록 static PSI의 연산 복잡도가 더 높으므로 Large Asymmetric PSI의 연산 효율을 더 크게 가진다. 발신 노드의 전처리 연산 시간 또한 X의 양이 대량일수록 static PSI의 연산 시간이 길게 소요되기 때문에 Large Asymmetric PSI의 연산 효율을 더 크게 가진다. tag 연산 과정에서는 tag생성 시간은 데이터의 양에 영향을 받지 않지만, tag 복구 시간은 데이터의 양에 따라 증가하는 특징을 가진다. 결과적으로 tag로 인한 연산 효율성을 입증하였다.

참고문헌

[1] Das, A., Bonneau, J., Caesar, M., Borisov, N., & Wang, X. (2014, February). The tangled web of password reuse. In NDSS (Vol. 14, No. 2014, pp. 23-26).

[2] Pearman, S., Thomas, J., Naeini, P. E., Habib, H., Bauer, L., Christin, N., Cranor. L. F., Egelman, S., Forget, A. (2017, October). Let's go in for a closer look: Observing passwords in their natural habitat. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 295-310).

[3] Zhang, Y., Monrose, F., & Reiter, M. K. (2010, October). The security of modern password expiration: An algorithmic framework and empirical analysis. In Proceedings of the 17th ACM conference on Computer and communications security (pp. 176-186).

[4] Verizon Business.(2020), Data breach Investigation Report, Available: https://enterprise.verizon.com/resources/reports/2020/2020-data-breach-investigations-report.pdf

[5] Pal, B., Daniel, T., Chatterjee, R., & Ristenpart, T. (2019, May). Beyond credential stuffing: Password similarity models using neural networks. In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 417-434). IEEE.

[6] Li, L., Pal, B., Ali, J., Sullivan, N., Chatterjee, R., & Ristenpart, T. (2019, November). Protocols for checking compromised credentials. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (pp. 1387-1403).

[7] Hagen, C., Weinert, C., Sendner, C., Dmitrienko, A., & Schneider, T. (2020). All the numbers are US: large-scale abuse of contact discovery in mobile messengers. Cryptology ePrint Archive.

[8] Trieu, N., Shehata, K., Saxena, P., Shokri, R., &Song, D. (2020). Epione: Lightweight contact tracing with strong privacy. arXiv preprint arXiv:2004.13293.

[9] Brickell, J., Porter, D. E., Shmatikov, V., & Witchel, E. (2007, October). Privacy-preserving remote diagnostics. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 498-507).

[10] Ion, M., Kreuter, B., Nergiz, E., Patel, S., Saxena, S., Seth, K., ... & Yung, M. (2017). Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive.

[11] Meadows, C. (1986, April). A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In 1986 IEEE Symposium on Security and Privacy (pp. 134-134). IEEE.

[12] Kiss, A., Liu, J., Schneider, T., Asokan, N.,&Pinkas, B. (2017, October). Private set intersection for unequal set sizes with mobile applications. In Privacy Enhancing Technologies Symposium(pp. 177-197). De Gruyter.

[13] Douceur, J. R., Adya, A., Bolosky, W. J., Simon, P., & Theimer, M. (2002, July). Reclaiming space from duplicate files in a serverless distributed file system. In Proceedings 22nd international conference on distributed computing systems (pp. 617-624). IEEE.

[14] Bellare, M., Keelveedhi, S., & Ristenpart, T. (2013, May). Message-locked encryption and secure deduplication. In Annual international conference on the theory and applications of cryptographic techniques (pp. 296-312). Berlin, Heidelberg: Springer Berlin Heidelberg.

[15] Wang, D., Zhang, Z., Wang, P., Yan, J., & Huang, X. (2016, October). Targeted online password guessing: An underestimated threat. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 1242-1254).

[16] HIBP, Available: https://haveibeenpwned.com/

[17] Buttyan, L. (2020, December). The Cost of Having Been Pwned: A Security Service Provider's Perspective. In Emerging Technologies for Authorization and Authentication: Third International Workshop, ETAA 2020, Guildford, UK, September 18, 2020, Proceedings (Vol. 12515, p. 154). Springer Nature.

[18] ENZONIC, Available: https://www.enzoic.com/

[19] Thomas, K., Pullman, J., Yeo, K., Raghunathan, A., Kelley, P. G., Invernizzi, L., ... & Bursztein, E. (2019). Protecting accounts from credential stuffing with password breach alerting. In 28th USENIX Security Symposium (USENIX Security 19) (pp. 1556-1571).

[20] Benny, P., Thomas, S., & Michael, Z. (2014). Faster private set intersection based on OT extension. In Usenix Security (pp. 797-812).

[21] Kolesnikov, V., Kumaresan, R., Rosulek, M., & Trieu, N. (2016, October). Efficient batched oblivious PRF with applications to private set intersection. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 818-829).

[22] Kannepalli, S., Laine, K., & Moreno, R. C. (2021). Password monitor: Safeguarding passwords in microsoft edge. Microsoft Research Blog, Available: https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/

[23] Chen, H., Laine, K., & Rindal, P. (2017, October). Fast private set intersection from homomorphic encryption. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1243-1255).

[24] Chen, H., Huang, Z., Laine, K., & Rindal, P. (2018, October). Labeled PSI from fully homomorphic encryption with malicious security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 1223-1237).

[25] Yu, X., Tang, D., Zhao, Z., & Zhao, W. (2024). Privacy-preserving compromised credential checking protocol for account protection. Computer Standards & Interfaces, 89, 103823.

[26] Kolesnikov, V., Kumaresan, R., Rosulek, M., & Trieu, N. (2016, October). Efficient batched oblivious PRF with applications to private set intersection. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 818-829).

[27] Pal, B., Islam, M., Bohuk, M. S., Sullivan, N., Valenta, L., Whalen, T., Wood, C., Ristenpart. T., Chatterjee, R. (2022). Might I get pwned: A second generation compromised credential checking service. In 31st USENIX Security Symposium (USENIX Security 22) (pp. 1831-1848).

[28] Li, J., Liu, Y., & Wu, S. (2021, May). Pipa: Privacy-preserving password checkup via homomorphic encryption. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (pp. 242-251).

[29] Gunther, D. (2019). Optimizing Private Information Retrieval for Compromised Credential Checking.

[30] Abadi, A., Dong, C., Murdoch, S. J., & Terzis, S. (2022, May). Multi-party updatable delegated private set intersection. In International Conference on Financial Cryptography and Data Security (pp. 100-119). Cham: Springer International Publishing.

[31] Badrinarayanan, S., Miao, P., & Xie, T. (2022). Updatable private set intersection. Proceedings on Privacy Enhancing Technologies, 2022(2).

[32] WIRED (2020), Security News This Week: 15 Billion Stolen Logins Are Circulating on the Dark Web, Available: https://www.wired.com/story/dark-web-credentials-roger-stone-blueleaks/#:~:text=It's%20no%20secret%20that%20hacker,passwords%E2%80%94stemming%20from%20100%2C000%20breaches.

[33] Cybernews (2021), RockYou2021: largest password compilation of all time leaked online with 8.4 billion entries, Available: https://cybernews.com/security/rockyou2021-alltime-largest-password-compilation-leaked/

[34] Verizon Business.(2021), Data breach Investigation Report, Available: https://www.idagent.com/blog/dark-web-data-fuels-credential-stuffing-trouble-for-1-1-million-accounts/

[35] Ponemon Institute (2017), The cost of Credential stuffing, Available: https://library.idgcommunications.net/idgcampaigns/documents/uploaded_data/d40/97b/1c-/original/the-cost-of-credential-stuffing_EN.pdf

[36] https://github.com/microsoft/APSI

[37] 동적 환경에서 Updatable Private Set Intersection을 이용한 크리덴셜 스터핑 공격 대응방안 연구", presented at Proceedings of the Annual Symposium of KIPS, Pyeongchang, Korea May, 23-25, 2024.

ABSTRACT

A Large Asymmetric PSI Computation Method for Mitigating Credential Stuffing Attacks

Geehee Yun
Department of Future Convergence Technology Engineering
Graduate School of Sungshin University

The Compromised Credential Checking(C3) service has emerged as a countermeasure against credential stuffing attacks, where compromised account information is is used to gain unauthorized access. Service providers collect leaked credentials to build a database, allowing users to request a comparison operation to check if their credentials have been compromised. This operation employs the Private Set Intersection (PSI) protocol to identify common data between both parties.

Traditional research on asymmetric PSI protocols has primarily focused on improving the efficiency and security of operations with asymmetric data sets. However, the increasing circulation of compromised accounts necessitates continual updates to the C3 server's data, requiring solutions to mitigate the computational and communication complexity caused by redundant operations. Additionally, the growing volume of data involved in these operations demands efficient handling of large data sets.

This study proposes and demonstrates a Large Asymmetric PSI operation method that enhances efficiency in asymmetric PSI operations involving large data sets. Our findings indicate that when

the sender possesses data from three existing asymmetric environments, 2^{20}, 2^{22}, 2^{24} and adds data of 2^{14}, 2^{16}, 2^{18} respectively, from additional operations, the larger the data set |X|, the greater the computational efficiency of the Large Asymmetric PSI. This efficiency compensates for the time required for data generation and recovery, optimizing overall operation time.

ACKNOWLEDGEMENT

본 논문은 한국정보처리학회 하계학술대회에서 발표한 ‘동적 환경에서 Updatable Private Set Intersection을 이용한 크리덴셜 스터핑 공격 대응방안 연구[37]’ 논문의 후속 연구로 수행되었습니다.

2년의 수학기간 동안 양질의 수업과 적극적으로 학술 활동에 참여할 수 있게 지원해주신 융합보안공학과 교수님들, 수업 프로젝트 팀원분들, 용역과제를 함께 수행한 선후배님들의 이끌어주심에 다 전하지 못한 감사의 인사를 남깁니다. 특히, 다양한 경험을 할 수 있도록 아낌없이 지원해주시고, 본 논문을 지도해주신 김경진 교수님과 귀한 시간 내어 논문을 발전시킬 수 있도록 도움주신 이주희 교수님께 감사드립니다.