동적 환경에서 Updatable Private Set Intersection 을 이용한 크리덴셜 스터핑 공격 대응방안 연구

윤지희 [1], 김경진 [2]
[1] 성신여자대학교 미래융합기술공학과 석사과정
[2] 성신여자대학교 융합보안공학과 교수

geehee_yun@naver.com, kyongjin@sungshin.ac.kr

Research on Countermeasures for Credential Stuffing Attacks Using Updateable Private Set Intersection in Dynamic Environments

Gee-Hee Yun[1], Kyoung-jin Kim[2]
[1]Dept. of Future Convergence Technology Engineering, Sungshin Women's University
[2]Dept. of Convergence Security Engineering, Sungshin Women's University

요 약

크리덴셜 스터핑 공격에 대응하기 위해 일부 기업에서는 C3(Compromised Credential Checking) 서비스를 제공한다. 인증 정보 대상 공격이 고도화됨에 따라 유출 계정의 양은 매우 방대한 것으로 드러나고 있으며, C3 서비스 서버의 데이터의 양 또한 지속해서 증가할 것이다. 그러나 C3 서비스의 PSI(Private set intersection) 프로토콜은 정적인 환경에서의 데이터 연산을 전제로 적용되고 있어 이용자가 반복연산을 요청했을 때의 연산 복잡도를 상쇄할 수단이 없다. 본 연구에서는 반복연산에 적용할 수 있는 updatable PSI 를 제안하였으며, static PSI 대비 x2~x4.3 의 전처리 시간과 x1.8~ x3.3 의 데이터 전송량이 효율적으로 개선되었음을 보인다.

1. 서론

정보통신기술의 발달과 함께 다양한 웹 · 앱 서비스가 등장하였고, 서비스는 사용자에게 아이디와 패스워드 기반의 인증 방법을 제공한다. 사용자는 지식기반 인증 방법인 로그인을 통해 간편하게 접근권한을 획득할 수 있다. 이에 따라 공격자에게도 사용자 인증 정보는 용이한 접근권한의 획득 수단이 되었으며, 악성코드, 피싱 등의 방법으로 계정을 탈취해 크리덴셜 스터핑(Credential stuffing) 공격에 활용함으로써 2 차 피해를 야기한다.

이에 대응하기 위해 C3(Compromised Credential Checking) 서비스가 등장하였다. 이 서비스는 사용자의 계정 입력값과 유출된 자격증명 기반의 DB 와의 비교 연산을 통해 자격증명 유출 여부를 확인하여 사용자 스스로 계정정보를 변경하도록 한다. 이때 PSI(Private set intersection) 프로토콜을 이용해 교집합 원소 이외의 정보 노출 없는 연산을 수행한다. 이는 상호작용 기반의 암호 프로토콜로 두 데이터 셋을 입력값으로 그들의 교집합을 찾고, 연산에 참여하는 한쪽 당사자에게 교집합 값을 출력한다. 일반적으로 출력값을 얻는 당사자를 수신자(Receiver), 나머지 한쪽 당사자를 발신자(Sender)라고 한다.

C3 서비스는 유출 자격증명을 발신자 집합에 업데이트하여 지속해서 최신화하며, 수신자의 집합 크기가 매우 작고 발신자 집합이 매우 큰 비대칭한 상황에서 연산이 이루어지는 특징을 가진다. 따라서 대량의 데이터에서 소량의 데이터와의 비교 연산 과정의 연산 효율과 안전성이 중요하다. 그러나 기존의 비대칭 데이터상에서의 PSI 연산은 소량의 인풋 정보 일부를 이용해 대량의 서버 정보의 연관 데이터를 추출한다. 이 과정에서 사용자의 일부 정보가 서버에 노출됨으로써 이를 공격자의 예측 예산을 감소에 이용하는 경우 위험성이 존재한다. 또한, PSI 프로토콜은 데이터가 고정된 일회성 연산의 효율성을 높이기 위한 연구가 주로 이루어진다. 따라서 기존의 PSI 를 C3 서비스에 적용함으로 낭비되는 통신 복잡도 상쇄 방안이 필요하다.

2. 선행연구 조사

2.1 비대칭 PSI 프로토콜

기존의 PSI 프로토콜은 수신자와 발신자가 유사한 연산 능력과 데이터 셋 크기를 가지는 것을 전제로 설계되었다. 이를 데이터 셋의 크기와 연산 환경이 불균형한 환경에 적용했을 때 통신 및 연산 복잡도 측면에서 성능 열화가 발생한다. 따라서, 제한된 자원과 인풋을 가진 수신자와 대량의 자원과 데이터를 보유한 발신자 간에 통신 및 연산 효율성을 높이기 위한 선행연구로 발전하였다.

비대칭 데이터 셋 환경에서의 PSI 프로토콜[1]은 OT(Oblivious Transfer) extension 을 기반으로 Bloom filter 를 통해 병렬화하여 연산 속도를 개선하였다. 이후, [2]는 Bloom filter 의 false-positive 응답률로 인한 문제 개선을 위해 cut and choose approach 방법으로 해결하고 낮은 오버헤드를 보장한다. 기존의 해싱 값 원소의 길이에 의존하는 방식에서 [3, 4], 이를 개선해 bins 에 매핑 시 항목의 길이를 줄임으로써 충돌을 줄이는 Permutation-based Hashing Set Intersection[5]을 이용하거나, 원소의 비트 길이에 의존하는 방식을 개선하면서도 약 3 배 이상 빠른 속도를 보장하는 방법을 제안했다[6]. 그러나, 소규모 데이터를 보유한 수신자는 대규모 데이터 셋에서 일부 데이터만을 추출하는 방법 중 하나로 hash-prefix 값으로 발신자의 hash 버킷 ID 를 추출해 버킷 내의 데이터로만 연산을 수행하는데, hash-prefix 가 서버에 노출되는 문제가 발생한다 [7]. [8, 9]는 Oblivious pseudorandom function(OPRF) 연산과 완전 동형암호화를 통해 이용자의 데이터가 서버에 노출되는 문제를 개선하고 연산 속도를 향상시켰다.

2.2 업데이트 가능한 PSI 프로토콜

PSI 프로토콜은 정적인 환경에서 일회성 연산의 효율성을 높이기 위한 연구가 대다수이다. 연산 대상 데이터를 업데이트하는 동적 환경의 PSI[10]는 Diffie Hellman 알고리즘 기반의 프로토콜로 소량의 데이터에 한정한 성능을 발휘한다. 또한 본 연구에서 제한하는 추가 및 삭제 방안은 적용되는 기술과 적용할 수 있는 환경이 한정되어 있어 적용 환경 최적화 연구가 추가로 요구되며, 총집합 크기가 충분히 크고 새로운 업데이트가 충분히 작은 상황에서 제한적으로 성능 효율을 보이는 한계점을 가진다. 따라서 대량의 데이터 업데이트 환경에서도 효과성을 가지는 업데이트 연산 기능이 요구된다.

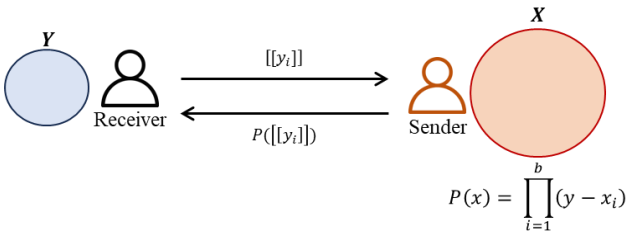
이처럼 교집합 연산의 안전성을 향상하는 완전 동형암호 기반의 프로토콜의 안전성을 유지하되 연산에서 발생하는 지연도를 상쇄할 수 있는 방안이 필요하다. 또한 고정된 데이터를 기반으로 한 PSI 프로토콜을 C3 서비스에 적용 시 중복 연산으로 연산량의 낭비가 발생한다. 따라서 발신자의 데이터가 정기적으로 업데이트될 경우 적용 관점에서 안전성과 효율성을 유지하며 업데이트할 수 있는 PSI 프로토콜이 필요하다.

3. 완전 동형암호 기반의 비대칭 PSI 프로토콜 분석

비대칭 PSI 프로토콜은 수신자 데이터의 일부를 발신자에게 노출해 대량의 데이터 셋에서 연관된 데이터 셋을 추출한다. 그러나, PSI 를 이용한 C3 서비스에서 수신자의 계정정보 일부의 노출은 공격자의 예측 예산을 대폭 줄여주는 위험이 있다. 따라서 본 연구에서는 동형암호 기반의 PSI 연산을 이용하여 프라이버시 향상을 가져온다. 다음은 동형암호 기반의 비대칭 연산을 간단히 설명하고, Hao Chen at al [9]의 Private Set Intersection 을 C3 서비스에 적용하는 방안을 설명한다.

3.1 동형암호 기반의 비대칭 Private Set Intersection

비대칭 PSI 연산에서 양자는 σ-비트의 문자열 집합을 보유하고 있으며, 발신자는 N_x 크기의 데이터 X 를, 수신자는 N_y 크기의 데이터 Y 를 보유할 때, $N_x > N_y$ 로 수신자가 모바일 장치와 같은 계산능력이 제한된 경우에 유용하게 동작한다. 동형암호 기반의 PSI 프로토콜은 (그림 1)로 표현하였다. 수신자가 Y 집합을 암호화하여 발신자에게 보내고, 발신자는 Y 집합에서 X 집합의 원소를 제거하는 비교연산을 수행하여 출력을 수신자에게 보낸다. 수신자는 복호화를 통해 교집합의 원소를 확인한다. 이때, $y \in X$ 일 경우에는 0 값을, $y \notin X$ 일 경우에는 원소 값을 출력하여 연산 결과 발신자에게는 아무것도 노출하지 않고 수신자만 $X \cap Y$ 를 확인한다.



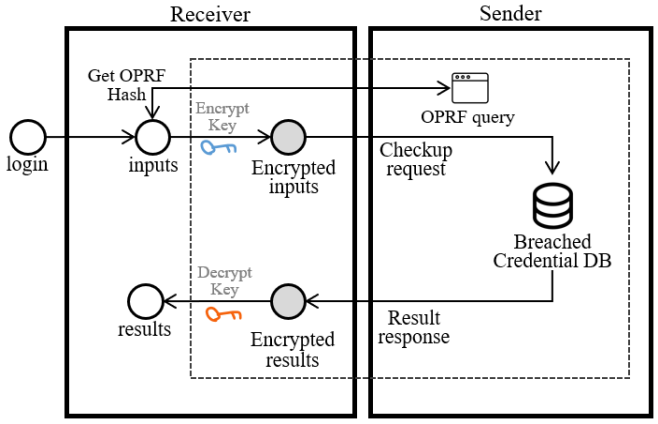
(그림 1) 동형암호 기반의 비대칭 PSI

3.2 Hao Chen at al 의 Private Set Intersection

Hao Chen at al [9]의 완전동형암호 기반의 PSI 는 P_0 과 P_1 이 보유 데이터의 양은 공개되어 있다는 전제하에

실행된다. 수신자와 발신자는 각각 P_0과 P_1이며, N_y 크기의 데이터 셋 Y, N_x 크기의 데이터 X를 보유한다. 전 처리 단계에서 P_1 은 적절한 PSI 파라미터와 Oblivious PRF 키 k 를 생성하고, 보유 데이터 셋 X에 대해 $X = \{OPRF_k(X_{new}) : x \in X\}$로 OPRF 연산한다. 파라미터 m은 cuckoo hashing 으로 N_y개의 공을 m개의 바구니 매핑에 이용한다. 예를 들어 랜덤한 3 개의 해쉬함수 $h_1, h_2, h_3 : \{0,1\}^\sigma \rightarrow [m]$ 이고, 모든 $x \in X'$ 를 $B[h_1(x)], B[h_2(x)], B[h_3(x)]$ 에 삽입한다. 이처럼 P_1은 전처리를 통해 연산비용을 절약한다.

P_0은 P_1에게 파라미터를 응답 받고, P_1이 Y를 OPRF 연산하여 $Y = \{OPRF_k(Y) : y \in Y\}$를 응답 받는다. 이때, OPRF 는 P_1만이 키를 알고 있으며, 연산 input 은 송신자만 알 수 있어 보안이 유지되는 알고리즘이다. P_0은 Y를 cuckoo hashing 하여 m개의 bins 로 구성된 테이블을 구성한다. 해당 원소를 FHE.Encrypt 하여 P_1에게 전송하고 P_1은 이를 Batching, Splitting, Windowing 하여 교집합 연산을 수행한다. P_0은 결과값을 복호화하여 교집합 원소의 존재 여부를 확인한다.

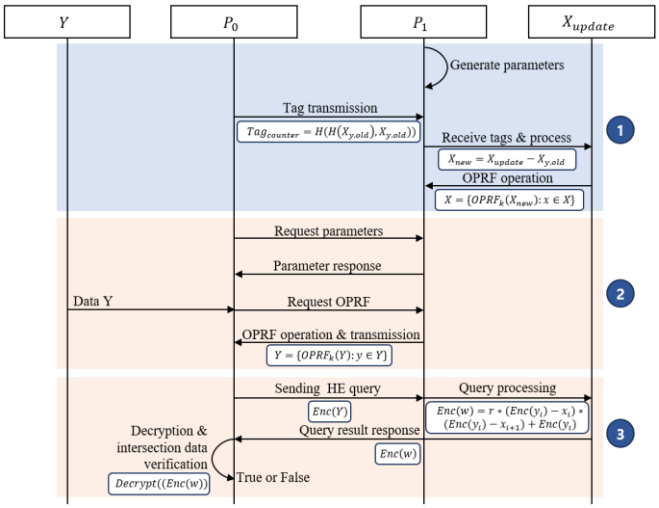


(그림 2) 동형암호 기반의 C3 서비스의 동작 구성도

4. Updatable PSI 제안

기존의 PSI 는 양자가 보유한 데이터 셋이 정적인 상태에서의 효율성을 높인다. C3 서비스의 특성상 1 회 연산에서 유출 계정을 확인하지 못할 경우 재연산이 필요하다. 최근 MOAB(Mother of all Breaches) 사건에서 유출 계정의 양을 260 억 개[11]로 드러날 정도로 업데이트 되는 계정의 양 또한 매우 크기 때문에, 재연산 시 낭비되는 연산량 또한 상당할 것이다. 기존에는 최초 연산에서 $X \cap Y = \emptyset$이었던 P_0이 재연산 시 P_1은 기존의 데이터 X_{old}와 새롭게 추가된 데이터 X_{new}를 합한 $X_{up} = X_{old} + X_{new}$ 에 대한 연산을 수행한다. 이는 X_{old}의 데이터로 발생하는 연산 및 통신 효율 손실이 발생한다.

(그림 3)은 업데이트된 발신자의 셋만큼 연산하는 Updatable PSI 이다. 1 단계, 재연산 시 공유하고 있는 다회성 연산 Tag 를 P_1에게 전송한다. 이때 Tag 는 P_1이 P_0의 연산요청 횟수인 counter 값에 따라 생성하며, $X_{y,old}$ 기반으로 암호키를 생성, 암호문 C 를 생성한 후 해쉬 함수를 이용해 H(C)의 Tag 를 생성해 사전에 전달한다. P_1은 보유한 Tag list 와 비교하여 중복연산 방지를 위해 $X_{up} - X_{old} = X_{new}$ 를 도출한다. 이에 대해 OPRF 를 이용한 전처리 연산을 수행한다. 2 단계, P_0은 P_1에게 파라미터를 공유받고, Y 에 대한 OPRF 연산을 요청하고 응답받는다. 3 단계, Y 에 대한 동형암호화 쿼리를 생성해 P_1에게 보유데이터와의 비교연산을 요청한다. 교집합 원소를 확인하기 위한 비교연산을 수행하고 암호화된 결과 $Enc(w)$ 값을 P_0에게 전송한다. 이를 복호화하여 연산 결과를 확인한다. 결과적으로 P_0은 X_{new} 에 대한 연산만을 응답 받는다.



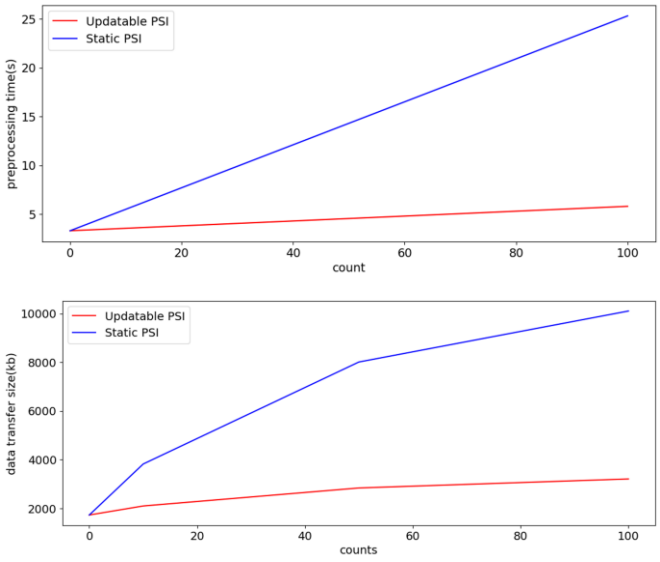
(그림 3) 업데이트 가능한 PSI 프로토콜

5. 실험

본 연구에서는 Ubuntu 20.04.6 LTS 환경에서 APSI 라이브러리[12]를 실행하였다. 실험에서는 발신자가 보유한 데이터 |X|는 2^{20}으로 지정하여 지속해서 데이터를 수집하고, 1 회에서 100 회까지 반복연산으로 교집합을 찾는다. |X|는 연산 횟수마다 이전 보유량의 10%씩 증가한다. 수신자가 보유한 데이터 |Y|는 1024 로 지정하고 고정하였다. 해당 환경에서 Static PSI 와 Updatable PSI 를 각각 실행하여 발신자의 전처리 시간과 수신자로의 데이터 전송량을 비교하였다.

(그림 4) (상)은 발신자의 전처리 시간을 나타낸다. 다량의 데이터를 가진 발신자는 연산 시간을 줄이기 위해 보유한 데이터에 대해 전처리 연산을 수행한다. Static PSI 는 증가하는 데이터의 양과 함께 중복 연산을 수행함에 따라 연산 시간이 선형적으로 계속 증가하고 있으나, Updatable PSI 는 업데이트되는 양만큼의 전처리 시간이 소요되어 x2~x4.3 의 효율을 가진다. (그림 4) (하)는 발신자의 수신자로의 데이터 전송량을 나타낸다. Static PSI 는 데이터양의 증가와 함께 중복 연산 되는 전송량을 반복적으로 처리함에 따라 전송량이 증가하나, Updatable PSI 는 제한된 증가 폭을 유지함으로써 x1.8~ x3.3 의 효율을 보인다.

결과적으로 Static PSI 에서 발신자의 전처리 시간과 통신량은 연산 횟수의 증가와 함께 전체 데이터양에 비례하여 증가하고 있음을 확인하였으며, 데이터의 양이 증가할수록 서버의 부담이 증가한다. 반면 Updatable PSI 는 증가하는 데이터의 양만큼의 제한된 영향을 받기 때문에 연산 횟수의 증가에도 제한된 전처리 시간과 통신량 증가 폭을 가진다.



(그림 4) |X|=2^{20}, |Y|=1024 인 경우 전처리 연산 시간(상) 및 통신량(하) 비교

6. 결론

본 연구에서는 동적 환경에서 Updateable Private Set Intersection 을 이용한 크리덴셜 스터핑 공격 대응방안을 제시하였다. C3 서비스는 특히, 사이버 공격의 초기단계에서 빈번하게 이용되는 크리덴셜 스터핑 공격을 방지할 목적으로 제공된다. 그러나 다량의 유출 계정이 업데이트 되고, 다수의 사용자가 반복적으로 서비스를 이용할 경우 서버의 연산 부담이 다량 발생한다. 따라서 업데이트 가능한 PSI 를 바탕으로 C3 서비스의 연산 부담을 감소시키고 이에 따라 연산 시간을 단축하는 방안을 제안하였다. 실험 결과 제안기술을 이용하면 x2~x4.3 의 전처리 시간 효율과 x1.8~ x3.3 의 서버의 통신량 효율을 가질 수 있다.

이를 통해 C3 서비스 제공자는 불필요한 연산 부담을 덜 수 있고, 서비스 이용자는 반복적인 연산을 통해 빠른 시간 안에 주체적으로 크리덴셜 스터핑에 대응할 수 있다. 향후 연구에서는 업데이트 기능을 통한 수신자와 발신자의 통신비용을 포함해 확인하고, 수신자의 인풋 데이터인 로그인 인증 정보의 변경이 가능한 updatable PSI 연산 방안을 구현 적용할 예정이다.

사사

본 논문은 2024 년도 산업통상자원부 및 한국산업기술진흥원의 산업혁신인재성장지원사업 (RS-2024-00415520)과 과학기술정보통신부 및 정보통신기획평가원의 ICT 혁신인재 4.0 사업의 연구결과로 수행되었음 (No. IITP-2022-RS-2022-00156310)

참고문헌

[1] Dong, C., Chen, L., & Wen, Z. (2013, November). When private set intersection meets big data: an efficient and scalable protocol. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 789-800).

[2] Rindal, P., & Rosulek, M. (2017, April). Improved private set intersection against malicious adversaries. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 235-259). Cham: Springer International Publishing.

[3] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, 24th USENIX Security Symposium, USENIX Security 15, pages 515–530, 2015.

[4] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. Cryptology ePrint Archive, Report 2016/930, 2016. http://eprint.iacr. org/2016/930.

[5] Pinkas, B., Schneider, T., Segev, G., & Zohner, M. (2015). Phasing: Private set intersection using permutation-based hashing. In 24th USENIX Security Symposium (USENIX Security 15) (pp. 515-530).

[6] 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).

[7] 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).

[8] 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).

[9] 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).

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

[11] https://cybernews.com/security/billions-passwords-credentials-leaked-mother-of-all-breaches/

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