Journal of The Korea Institute of Information Security & Cryptology(JKIISC) ISSN 1598-3986(Print)
VOL.23, NO.4, August 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.4.629

개인정보 보호를 위한 데이터의 자가 초기화에 대한 고찰*

김 종 욱,† 강 석 인, 홍 만 표‡
아주대학교

A Study on the Self-destructing Data for Information Privacy*

Jonguk Kim,† Sukin Kang, Manpyo Hong‡
Ajou University

요 약

최근 개인정보 보호에 대한 관심이 높다. 디지털로 변환된 정보는 인터넷을 통해 쉽게 전달되고 개인별 맞춤 서비스를 위해 서비스 제공자가 사용자의 개인정보를 요구하는 경우가 많아졌다. 사용자들은 서비스 제공자들에게 정보를 제공할 때 암호화를 통해 안전하게 전달하면 공유된 개인 정보가 안전하게 보호된다고 판단하고 있으나 서비스 제공자가 활용한 개인 정보를 정확한 시점에 파기하지 않고 필요 이상으로 오랫동안 보관함으로써 정보 유출의 가능성을 높이는 경우가 많다. 개인정보를 필요한 기간만큼만 보관하고 필요가 없을 경우 삭제하거나 초기화 한다면 개인정보의 유출 가능성을 훨씬 낮출 수 있다. 본 논문에서는 개인정보 보호를 위한 데이터 자가 초기화 기법에 대해 고찰하고 문제 해결을 위해 유리 상자 모델을 제안한다.

ABSTRACT

Recently the interest in the information privacy has been growing. Digital data can be easily transferred via Internet. Service providers ask users for private data to give customized services. Users believe that their shared data are protected as they deliver their private data securely. However, their private data may be leaked if service providers do not delete or initialize them when they expire. The possibility of information leak may lower if the service providers deal with users' private data properly. In this paper, we study the self-destruction of private data for information privacy and propose the *glass-box* model.

Keywords: Information Privacy, Information Security, Digital Forensic, Data Privacy Model

I. 서 론

현대 정보화 사회에서 정보의 활용도가 증대될수록 개인정보보호의 침해에 대한 두려움과 함께 개인 정보를 보장할 수 있는 방법에 대한 관심이 높다. 보안의 세 가지 요소로 불리는 기밀성, 무결성, 가용성에 개인정보보호를 추가해야 한다는 주장도 설득력을 얻고 있다. 유비쿼터스 컴퓨팅에 대한 연구가 활발하게 진행되면서 사용자의 상황 인식을 위해 개인 정보를 수집하게 되고, 모바일 기기의 사용이 일반화되면서 사용자 위치나 행동 패턴 등이 암암리에 파악되기도 한다. 네트워크의 접근성, 연결성, 전송속도의 개선으로 클라우드 컴퓨팅 환경이 보편화되어 사용자들은 이제 자신이 보유하고 있는 정보 중 많은 부분을 클라우드

접수일(2013년 7월 1일), 수정일(2013년 7월 26일), 게재확정일(2013년 7월 27일)
* 본 연구는 산업통산자원부/미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업의 일환으로 수행하였음. [10041865, 클라우드 서버와 스마트 기기간의 암호화된 콘텐츠 실시간(1Giga bps급) 동기화 소프트웨어 개발]
† 주저자, kju@ajou.ac.kr
‡ 교신저자, mphong@ajou.ac.kr(Corresponding author)

에 보관하고 있다. 현재의 컴퓨팅 환경은 개인정보 침해 가능성에 대한 우려를 가질 수밖에 없는 상황이다.

기밀성과 무결성은 암호 기술을 통해서, 가용성은 물리적 보안과 침입 탐지 시스템을 포함한 다양한 보안도구들을 통해 어느 정도 해결하고 있으나 개인정보 보호 문제는 아직까지 해결이 쉽지 않다. 문제의 해결을 어렵게 만드는 점 중에 하나는 개인정보가 침해되었다고 생각하는 기준이 개인 또는 사회마다 다르기 때문이다[1]. 어떤 문화에서는 당연히 개방되는 것이 어떤 문화에서는 상당히 개인적인 것으로 받아들여지기도 한다. 개인정보보호에 대해서는 많은 정의가 존재하지만 그 중 하나로 "정보 노출 대한 제어(privacy is control over information disclosure)"라는 것이 있다. 정보를 어느 정도로 노출시킬 것인가는 결국 정보의 주인이 결정해야 하는 것임을 의미하는 말이기도 하다. 즉, 현재까지의 개인정보 보호 문제는 정보 소유자의 '제어'에 초점이 맞추어져 왔고, 따라서 정책(policy)을 이용해서 개인정보를 보호하려는 시도가 많았다. 정보는 수동적인 것이며, 누가 접근을 할 수 있는 권한을 가지는 지에 대한 여부, 즉 접근 제어를 이용한 해결이 주를 이루어왔다. 그러나 전통적인 접근 제어만으로는 개인정보보호 문제를 모두 해결하기 어렵다.

기존 보안의 세 가지 요소는 비교적 침해 여부의 판단을 확실히 할 수 있고 침해의 경로가 정해져 있다. 기밀성의 경우 제 삼자가 해당 정보의 정확한 내용에 접근하였는가 여부, 무결성의 경우 제 삼자가 해당 정보에 어떤 수정을 가했는가에 대한 여부, 가용성의 경우 사용자가 어떤 자원에 대해 정상적으로 접근할 수 있는지에 대한 여부로 판단이 가능하다. 그러나 개인정보의 경우 어떤 경우 침해가 되었다고 판단할지 명확히 정의된 바가 없다. 개인정보보호의 목표를 어떻게 설정하는가에 따라 보안과 개인정보보호는 우호적 관계가 되기도 하고 적대적인 관계가 되기도 한다[2].

본 논문에서는 혼재하는 개인정보보호에 대한 정의를 다음과 같이 네 가지로 정리한다. 네 가지 분류는 개인정보의 유출 경로를 기준으로 한 분류이다.

1. 정보 자체의 노출
2. 정보 교환 주체의 노출
3. 필요 이상의 정보가 전달
4. 정보가 필요한 곳에만 사용되지 않고 다른 곳에도 이용되는 것

첫 번째 문제는 전통적인 보안 요소인 기밀성과 연관성이 깊다. 주민등록번호나 주소 등 개인 정보가 저장된 데이터베이스가 노출되는 경우, 우리는 개인정보가 침해되었다고 말한다. 보안과 개인정보보호를 동일한 문제로 보는 관점은 이 문제에서 파생된다. 개인정보를 보관하는 데이터베이스에서 정보가 유출되지 않도록 하는 것은 보안성(기밀성)도 높이고 개인정보도 보호하기 때문이다.

두 번째 문제는 정보가 전달될 때 해당 내용도 중요하지만 누가 누구에게 전달하는지에 대한 정보 또한 중요함을 지적한다. 예를 들어 어떤 NGO 활동가가 위키리크스(WikiLeaks)에 중요한 비리에 대해 고발하려 할 때, 이와 같은 상황에서는 그가 고발하려 하는 내용과 무관하게 고발을 위해 누군가와 접촉을 시도했다는 것 자체가 신변에 위협을 가할 수 있다. 정보의 콘텐츠가 아니라 네트워킹에 참여하는 시점과 누구와 교신했는가에 대한 데이터 또한 개인 정보가 될 수 있다. 이 문제의 해결은 대부분 익명 통신(anonymous communication)을 통해 이루어진다. 익명 통신을 위한 기본적인 개념으로는 믹스 네트워크 (MIX-net)이 있다. 메시지를 다중 암호화하고 여러 번 전달함으로써 메시지의 최초 송신자와 최종 수신자를 알아내기 어렵게 만드는 방식으로, Chaum의 추적 불가능한 전자 우편 시스템[3]에서 시작했으며, 현재 대표적인 시스템으로는 Tor가 있다[4]. 익명 통신에 대한 상세한 설명은 본 논문에 포함하지 않는다. 익명성과 관련된 용어와 개념은 Pfitzmann 등[5]에 잘 정리되어 있다.

세 번째 문제는 첫 번째 문제와 대척점에 위치한다. 첫 번째 문제에서 언급한 개인정보가 저장된 데이터베이스는 인증을 위해 정보를 수집한 것으로 보안성을 높이기 위해 개인정보 노출 가능성을 높이는 결과를 초래한다. 개인정보보호를 위해서는 정보를 최소한으로 수집해야 하지만 잘 지켜지지 않고 있다. 국내 웹사이트에 가입할 때 우리는 많은 개인 정보(주민등록번호, 주소, 전화번호, 직업 등)를 입력하도록 강요받는다. 2012년 위헌 판정을 받은 '인터넷 실명제'의 시행은 국내 서비스 업체들이 비교적 많은 개인 정보를 요구하게 만들었고 사용자들도 개인정보를 제공하는 데 익숙해졌다. 그러나 개인 정보를 타인에게 전달함에 있어서 그 행위가 반드시 필요한 것인지를 반문해야 한다. 한 번 소유주를 떠나 타인에게 전달된 정보는 더 이상 통제하기가 어렵다. 정보의 최초 공개 시점이 중요하다. 개인 정보의 제공은 정보를 소유하고

있는 해당 개인이 결정하는 것이 가장 바람직하다. 예를 들어, 길 찾기 서비스를 받고 싶은 개인이라면 대략의 내 위치를 알려주어야만 서비스를 받을 수 있다. 어느 시점에 어느 정도의 정보를 제공할지는 개인의 몫이다. 길을 완전히 잃은 경우라면 정확한 내 위치를 알려주어야 할 경우도 발생한다. 필요한 만큼의 개인정보 전달은 P3P(Platform for Privacy Preferences)[6]와 같이 정책을 통해 개인 정보 제공 여부를 결정 할 수 있도록 하거나 서비스 제공자가 사용자 개인정보를 정확히 파악할 수 없도록 혼란스럽게 만드는 방법이 있다. 앞선 예에서와 같이 위치 정보를 전달하는 경우에는 정확한 서비스를 받기 위해 위치 정보를 알려주지 않을 수 없기 때문에 k-anonymity 등의 기법을 이용하여 위치를 은폐(location cloaking)함으로써 개인정보와 신원 간의 관계를 분리해내기도 한다[7].

네 번째 문제는 사람들에게 널리 인식되지 부분으로 본 논문이 해결하고자 하는 것이다. 개인정보의 침해는 정보가 파기되어야 할 시점에 파기되지 않고 계속 유지됨으로써 발생하는 경우가 많다. 우리는 서비스를 받기 위해 부득이 개인 정보를 타인에게 전달해야 하는 상황에 직면한다. 자동차가 주차장에 진입할 때 보안과 주차 요금 계산의 자동화를 위해 입구에서 차량 번호 인식을 한다고 가정해보자. 차량이 진입한 시점부터 주차 요금을 정산하고 주차장을 빠져나갈 때까지 자동차에 대한 정보는 당연히 유지되는 것이 옳으며 사용자도 불만을 가질 이유가 없다. 위에서 언급한 세 번째 문제 또한 잘 해결되어 정확히 필요한 정보만을 전달했다고 가정하자. 그럼에도 불구하고 사용자가 주차장을 빠져나간 이후에도 차량 번호가 주차장의 전산 시스템의 데이터베이스에 남아 있게 된다면 그것은 개인정보의 침해가 될 소지가 있다. 이후에 어떤 사용자가 주차장에 언제부터 언제까지 있었는지를 알려주는 자료로 남용될 수 있기 때문이다. 내 기기에 저장된 정보는 오래도록 안전하게 보관되어야 한다. 그러나 다른 사람에게 필요에 따라 전달된 내 정보는 더 이상 필요하지 않은 시점이 오면 즉시 파기되는 것이 옳다.

본 논문은 네 번째 문제에 대한 해결 방안을 제안한다. 첫 번째 문제는 암호화를 통한 기밀성 보장으로, 두 번째 문제는 익명 통신 기법으로, 세 번째 문제는 개인의 부주의한 정보 취급을 예방하고 감지함으로써 해결할 수 있다. 그러나 네 번째 문제에 대한 해결은 논의가 부족한 실정이며 해결 또한 쉽지 않다. 정보 공유가 불가피하게 선행된 상황에서 공유한 데이터에 대한 제어는 더 이상 불가능하다. 개인 정보의 주인이 공유한 정보에 대해 제어 권한을 유지할 수 있도록 하는 방안이 필요하다. 2장에서는 개인 정보의 무한한 재사용을 방지하기 위한 유리 상자 모델을 제안하며, 3장에서는 제안한 모델의 구현 방법을 강구한다. 4장에서는 유리 상자 모델을 회피할 수 있는 디지털 콘텐츠의 복제 문제를 다루고 대응 방안을 제안한다. 5장에서는 관련 연구와의 차이점을 기술하며, 6장에서 논문을 결론짓는다.

II. 유리 상자 모델: 폭탄이 부착된 개인정보

정보를 누군가에게 전달할 때 개인 정보가 포함되어 있다면 해당 정보가 언젠가 파기되어야 한다는 보장이 있어야 한다. 정보의 파기는 두 가지 관점으로 접근할 수 있다.

우선 정보를 건넨 사람에게 반드시 파기한다는 약속을 받는 방법이 있다. 데이터는 기본적으로 수동적인 성격을 가진다. 내 손을 떠나 상대에게 전달된 정보는 상대가 파기해주는 방법밖에 없다. 이 접근법은 기술적인 해결은 어렵다. 상대방이 실제로 정보를 완전히 삭제했는지 확인할 수 있는 방법이 없기 때문이다. 상대방이 약속을 이행하지 않았음을 알게 되었을 때 법적으로 강력히 대응할 수 있겠으나 상대가 내 개인정보를 원칙대로 다루었는지 여부를 알아내는 것은 쉽지 않다.

두 번째 방법은 데이터가 능동적으로 자신을 파기시키는 것이다. 개념적으로는 정보에 폭탄(데이터를 초기화 할 수 있는 장치)을 부착하는 것이다. 정해진 시간이나 접근 회수 등의 조건에 따라 폭탄이 폭발함으로써 데이터가 초기화 될 수 있다면 개인정보의 불필요한 유통을 방어할 수 있을 것이다. 비슷한 개념으로 Stajano의 grenade timer가 있다[8]. Grenade timer는 원격에서 보내진 모바일 코드를 안전하게 실행할 수 있는 방안으로 모바일 코드가 악의적인 행동을 할 수 없도록 의심되는 코드의 동작을 즉시 멈추도록 할 수 이는 장치이다. 하지만 코드와 달리 정보(데이터)는 수동적이다. 정보만을 전달해서는 행위를 유발할 수 없다. 본 논문에서는 정보와 폭탄이 함께 담겨있는 '유리 상자'를 정의한다. 유리 상자는 다음과 같은 특징을 가진다.

조건 1. 투명성 (가독성)
조건 2. 무결성
조건 3. 자가 초기화
조건 4. 전달 가능

투명한 유리 상자는 누구든 내부를 들여다볼 수 있다. 따라서 누구든 유리 상자 내부에 있는 정보를 열람할 수 있다(조건 1). 단, 유리 상자를 열 수는 없다(조건 2). 유리 상자는 데이터를 다루는데 있어서 최소 단위가 된다. 유리 상자에는 폭탄이 함께 포장되어 있으며, 폭탄이 폭발하면 함께 있던 정보가 파괴된다(조건 3). 유리 상자를 열 수 없다는 것은 그 안에 함께 담겨진 폭탄을 제거할 수 없음을 의미한다. 유리 상자 내의 폭탄은 조건이 만족되면 폭발한다. 여기서의 조건은 시간이나 데이터 열람 회수 등이 활용될 수 있다. 그리고 유리 상자는 다른 누군가에게 전달될 수 있다 (조건 4). 전달 과정에서 유리 상자가 파괴되거나 전달 후에 변형되는 일은 발생하지 않는다. 개인정보를 다른 개체에게 전달하고 전달된 정보가 소비되는 과정을 정리하면 다음과 같다.

1. 정보의 주인은 해당 정보를 유리 상자에 담고 폭탄이 언제 어떻게 동작할지를 설정한 뒤 전송
2. 유리 상자를 전달 받은 수신자는 유리 상자에 담긴 데이터를 확인하고 이용
3. 수신자 측에 보관되어 있는 유리 상자는 조건에 따라 폭탄이 동작하게 되고 내부의 개인 정보는 적절히 파기됨

유리 상자 모델이 의도대로 개인정보를 보호하기 위해서는 다음과 같은 조건이 만족되어야 한다.

- 데이터와 폭탄을 함께 담고 있는 유리 상자를 실제로 구현할 수 있어야 함
- 중간자나 수신자가 유리 상자를 파괴하지 못하도록 방어할 수 있어야 함
- 수신자가 데이터를 확인한 후 복사본을 별도로 만들 경우에 대응할 수 있어야 함

우선 유리 상자 모델을 어떻게 구현할 수 있는지 설명한다. 유리 상자의 구현을 위해서는 코드와 데이터가 함께 강력히 묶여 있는(tightly-coupled) 무언가를 만들어야 하며 데이터와 코드가 결속된 개념은 이미 존재한다. 객체 지향(object-oriented concept)에서의 객체이다.

III. 구현 방안: 객체

객체는 데이터와 코드를 포함한 개념이며 객체에 포함된 코드(메서드, method)는 객체 내부의 데이터에만 접근한다. Blakley는 객체라는 개념이 데이터와 코드간의 경계를 모호하게 하여 기존에 가지고 있던 무결성 점검 도구들을 무력화시켜 버렸으며 보안에 악영향을 미쳤다고 언급한 바 있다[9]. 그러나 데이터와 코드를 분리할 수 없도록 결합시킴으로써 유리 상자 모델이 제안한 바와 같이 데이터에 자가 초기화를 위한 코드를 추가할 수 있게 된다. 객체를 이용하면 메시지를 수신한 쪽에서는 항상 특정 코드를 실행해야만 데이터에 접근할 수 있도록 강제할 수 있다. 일반적으로 객체의 데이터는 내부에서만 접근 가능하도록 숨기며 외부에서는 인터페이스를 통해서만 접근할 수 있도록 한다. 객체지향에서의 객체가 가지는 기본 성질인 캡슐화(encapsulation)다. 이를 이용하면 유리 상자의 조건 1, 2, 3을 모두 만족시킬 수 있다. 조건 4는 객체가 가지는 또 하나의 성질인 직렬화(serialization)를 이용한다. 직렬화는 컴퓨터 공학에서 데이터 저장과 전송에 관련된 개념으로써 어떤 자료 구조나 객체를 다른 컴퓨팅 환경으로 전달하기 위한 기술을 말한다. 객체 지향에서는 데이터와 코드의 보관 및 전달을 위해 직렬화를 제공하고 있다.

2장에서 정리한 유리 상자 모델을 객체지향으로 구현하면 다음과 같다.

1. 메시지 송신자는 객체를 생성하면서 내부 데이터에 전달하고자 하는 개인정보를 설정한다. 즉, 객체의 생성자(constructor)에서 전달하고자 하는 데이터를 받아 객체 내부의 데이터로 설정한다.
2. 내부 데이터를 읽을 수 있는 메서드 - 일반적으로 getter라고 불리는 - 코드를 구현한다. 이 메서드는 폭탄의 동작 조건과 폭탄의 실제 동작에 대한 코드를 포함한다. 내부 데이터를 설정할 수 있는 - 일반적으로 setter로 부르는 - 메서드는 정의하지 않는다. 처음 객체가 생성될 때 설정된 내부 데이터가 수정될 수 있는 방법은 없다.
3. 메시지는 객체를 직렬화한 것(serialized or marshalled object)을 전송한다.

```
interface GlassBox extends Serializable {
    public String get();
}
```

〔그림 1〕 유리 상자 객체를 위한 인터페이스

4. 메시지 수신자는 받은 객체에 정의된 메서드(getter)를 호출하여 데이터를 얻어온다.

유리 상자 모델을 객체로 구현할 수 있음을 보이기 위해 코드로 표현해본다. 객체 표현을 위해 의사 코드(pseudo code)를 사용하는 것이 바람직하나 객체 표현을 위한 표준 의사 코드가 없으므로 가장 널리 사용되는 자바(Java) 언어〔10〕로 표현한다. 예제에서는 구현의 편의와 간결성을 위해 접근 회수 제한 기법을 이용하는 자가 초기화 예제를 보인다.

〔그림 1〕의 GlassBox 인터페이스는 유리 상자 모델이 가지고 있어야 하는 데이터 접근 메서드와 객체의 직렬화를 위한 코드가 포함되어 있다. 〔그림 2〕의 BasicGlassBox 클래스는 GlassBox 인터페이스를 구현한 클래스이다. 간단한 예제를 위해 유리 상자 내에 저장되는 데이터는 소문자로만 구성된 영문자열로 가정한다. 몇 번 데이터가 읽혀졌는지를 기록할 변수(used)와 몇 번 읽혀졌을 때 초기화 될 것인지에 대한 임계값(limit)을 가진다. 외부에서는 get 메서드를 통해서만 GlassBox 객체에 저장되어 있는 문자열을 받아올 수 있다. 따라서 메시지를 읽을 때 특정 코드가 실행될 수 있도록 강제하는 효과를 얻는다. 〔그림 3〕은 BasicGlassBox 객체를 실제로 생성하여 데이터 초기화 작동을 점검하는 코드이다. 내부 데이터에 대해 두 번만 접근이 가능하도록 설정하였기 때문에 세 번째 사용하였을 때 데이터가 자동으로 초기화되었음을 확인할 수 있다. 객체를 이용하면 유리 상자 모델이 가져야 하는 조건 1, 2, 3을 만족시킬 수 있다. 조건 4에 대한 예는 4장에서 설명한다.

본 장에서는 유리 상자의 구현 가능성에 대해 제안하였다. 수신자가 유리 상자를 파괴할 가능성과 복사본을 통한 유리 상자 모델의 우회 가능성에 대해 추가적인 설명이 필요하다. 유리 상자 자체가 파괴되어 온전히 필요한 정보만을 담고 있는 데이터가 공개될 수 있는 가능성은 분명히 존재한다. 객체가 메모리나 파일에 담겨 있을 경우 운영체제 수준에서 데이터 자체(raw data)에 접근하게 되면 유리 상자 내부의 데이터에 직접 접근이 가능하거나 코드를 수정할 수 있게

```
public class BasicGlassBox implements GlassBox {
    protected String data;
    protected int used;
    protected int limit;

    public BasicGlassBox(String init, int times) {
        used = 0;
        data = init;
        limit = times;
    }

    @Override
    public String get() {
        if (++used > limit) data = "initialized";
        return data.toLowerCase();
    }
}
```

〔그림 2〕 유리 상자 구현을 위한 클래스 예 (영소문자열을 내용으로 담고 있으며, 사용 횟수에 의해 자가 초기화 되는 유리 상자)

된다. 이것은 객체가 인식될 가상 머신이나 혹은 해당 운영체제와 동일한 수준 또는 그 이하의 수준에서 접근하는 공격이므로 차단이 쉽지 않다. 본 논문에서는 모든 개인정보는 반드시 유리 상자에 담겨 전달되고 활용된다고 가정한다. 유리 상자에 담겨있지 않은 데이터는 모두 무효처리한다. 다시 말해서 공격자가 정교한 공격을 통해 데이터 자체를 얻어내었다고 해도 사용을 위해서는 다시 유리 상자에 담아야만 하도록 정책적으로 강제한다. 이것은 강력한 가정이며 전체 인터넷 환경에 적용하는 것은 쉽지 않으나 군이나 기업 등의 특정 그룹 내에서의 네트워크에서는 객체에 대한 정의를 통일하고 정보를 주고받는 모든 개체가 유리 상자를 이해해야 하도록 가정할 수 있다.

IV. 유리 상자의 복사에 대한 대응

천재적인 해커로 유명한 Kevin Mitnick은 얻고자 하는 정보가 있을 때 백업을 목표로 하는 것도 유용한 전략이라 말했다〔11〕. 일반적으로 백업은 기관의 지침에 의한 귀찮은 작업으로 여겨지고 백업된 데이터는 원본만큼이나 공을 들여 관리하지 않는 경우가 많다. 그러나 디지털 정보는 복사본과 원본의 차이가 없다. 복사본이 생성되었다면 항상 그 복사본에게도 동일한 보안 정책이 적용되어야 한다. 본 논문이 제시한 유리 상자 모델에서도 유리 상자를 복제할 경우 약점을 가질 수 있다. 원본이 폭탄에 의해서 제거되어도 여전히 - 원본과 완전히 동일한 - 복제된 정보가 존재할 수 있기 때문이다.

```
String privateData = "top secret.";
GlassBox basicGlassBox = new BasicGlassBox(privateData, 2);
System.out.println(basicGlassBox.get());
System.out.println(basicGlassBox.get());
System.out.println(basicGlassBox.get());

결과:
top secret.
top secret.
initialized
```

〔그림 3〕 BasicGlassBox 객체를 이용한 정보 자가 초기화 예제

해당 데이터를 암호화함으로써 해결할 수 있지 않은지 반문할 수 있다. 암호화는 유리 상자가 아니라 불투명한 검은 상자에 비유해볼 수 있는데, 암호화 역시 복사 문제를 해결할 수는 없다. 정상적인 메시지 수신자라면 암호 키를 가지고 있을 것이고 내용을 열어볼 수 있기 때문이다. Boneh는 암호화를 통해 보관된 데이터를 직접 삭제하지 않고도 무효화 시키는 아이디어를 제안한 바 있다〔12〕. 그러나 이것은 개인적인 백업 콘텐츠를 다루는 문제로 본 논문에서 가정하는 두 개체간의 합법적인 정보 전달 후의 데이터 처리 문제와는 목적이 다르다. 본 논문에서는 송신자 입장에서 수신자를 정상적인 사용자로 가정하므로 암호화 기법이 추가되더라도 수신자는 암호문을 복호화 할 수 있다. 즉, 내용을 열어본 수신자가 데이터를 이후에 공개할 수 있기 때문에 암호화는 문제를 근본적으로 해결하지 못한다. 다시 말해서, 암호화는 데이터에 접근할 수 있는 권한을 가지지 않은 중간자로부터 데이터를 보호할 수는 있으나 - 유리 상자 모델이 해결하고자 하는 - 정보를 합법적으로 전달 받은 측에서 데이터를 축적하는 것을 방지하는 문제를 해결하지는 못한다. 유리 상자가 복사되는 경우는 두 가지로 나누어볼 수 있다.

1. 유리 상자 자체가 복제되는 경우
2. 유리 상자가 가지고 있는 데이터를 이용하여 새로운 유리 상자를 생성하는 경우

유리 상자 자체가 복제되는 것은 내부의 폭탄도 함께 복제됨을 의미한다. 예를 들어, 폭탄의 작동 조건이 시간제한 방식인 경우 복제된 유리 상자가 함께 폭발할 것이므로 공격자에게 큰 이득을 줄 수 없지만 접근 회수 제한 방식의 폭탄인 경우에는 의미 있는 공격법이 된다. 시간제한의 경우에는 시간 동기화가 전제되어 있어야 하고 수신자 측에서 시스템의 시간 정보(system clock)를 조작할 수 없다는 가정이 필요하다. 폭탄의 작동 조건은 시간제한이나 접근 회수 제한에 국한되는 것이 아니라 시스템의 필요에 따라 설정할 수 있으며 이를 어떻게 설정하느냐에 따라 시스템이 가져야 할 전제 조건이 달라진다.

```
public class CopyDetectionGlassBox extends BasicGlassBox {
    private HashMap<String, String> history;
    private Random r;

    public CopyDetectionGlassBox(String init) {
        super(init, -1);
        r = new Random();
        history = new HashMap<String, String>();
    }

    public GlassBox pack(String toWhomAndWhen, int limit) {
        shuffle();
        history.put(this.data, toWhomAndWhen);
        return new BasicGlassBox(data, limit);
    }

    public String check(BasicGlassBox gb) {
        if (history.containsKey(gb.data)) return history.get(gb.data);
        return "Invalid glassbox!";
    }

    private void shuffle() {
        StringBuilder sb = new StringBuilder();
        for (char c : data.toCharArray())
            sb.append(r.nextBoolean() ?
                Character.toUpperCase(c) : Character.toLowerCase(c));
        data = sb.toString();
    }
}
```

〔그림 4〕 복제 감지를 위해 스테가노그라피를 추가한 객체 예 (get 메서드 외에 pack과 check 메서드 추가)

유리 상자의 데이터에 합법적으로 접근한 후 해당 데이터만을 복사하여 새로운 유리 상자에 포장함으로써 개인 정보를 계속 유지하는 공격이 있을 수 있다. 이 방법은 정보 소유자가 설정한 폭탄을 우회할 수 있다. 데이터에 복제를 차단할 수는 없기 때문에 원본과 복사본을 구분할 수 있는 방안이 필요하고 디지털 포렌식을 위해 복사되는 시점 또한 파악할 필요가 있다. 복제를 감지하기 위해 확장된 유리 상자 모델에서는 다음과 같이 데이터 복사를 추적한다.

- 개인정보의 주인은 언제나 해당 정보에 접근할 수 있으며 소유자의 접근에 의해 데이터가 자가 초기화 되지 않음
- 개인정보를 타인에게 전달하고자 할 때 유리 상자를 새로 생성하여 폭탄을 설치하고 전송
- 개인정보에는 누구도 예측할 수 없는 표시가 숨겨져 있으며 개인정보를 다른 개체에게 전달하기 위해 새 유리 상자에 담을 때마다 변화
- 모든 내부 데이터의 변화는 기록됨

공개된 개인정보의 유효성을 점검하고 비정상적인 데이터가 유통되고 있다면 이 데이터를 유통시킨 자를

```
String privateData = "top secret.";
CopyDetectionGlassBox alice = new CopyDetectionGlassBox(privateData);

GlassBox forEve = alice.pack("Eve " + new Date(), 2);
File tmpFile = File.createTempFile("obj_", ".tmp");
FileOutputStream fos = new FileOutputStream(tmpFile);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(forEve);

FileInputStream fis = new FileInputStream(tmpFile);
ObjectInputStream ois = new ObjectInputStream(fis);
GlassBox eve = (GlassBox) ois.readObject();

String receivedData = eve.get();
System.out.println(receivedData);
System.out.println(alice.check((BasicGlassBox) eve));

GlassBox mallory = new BasicGlassBox(receivedData, 2);
System.out.println(mallory.get());
System.out.println(alice.check((BasicGlassBox) mallory));

결과:
top secret.
Eve Thu Jun 20 00:22:56 KST 2013
top secret.
Invalid glassbox!
```

〔그림 5〕 직렬화를 통한 유리 상자의 전달 예와 유리 상자의 진위 확인 예

추적할 수 있어야 한다. 추적을 위해서는 유리 상자에 유리 상자를 생성한 자만이 알아볼 수 있는 표시가 필요하다. 복사 기록 추적을 위한 방안으로 숨겨진 표식(steganography)을 활용한다.

〔그림 4〕에서의 CopyDetectionGlassBox 클래스는 복제를 추적할 수 있는 방안을 간단히 예제로 표현하고 있다. 데이터는 소문자로만 구성된 문자열로 가정한다. 이 클래스는 원본의 보존을 위해 초기 데이터에 대해서는 폭탄을 설정하지 않는다. 이후에 다른 누군가에게 개인 정보를 전달해야 할 때 pack 메서드를 통해 새로운 GlassBox를 받는다. Shuffle 메서드에 의해 내부의 데이터가 수정되는데 임의의 위치에 글자를 대문자로 변경한다. 데이터를 받아오기 위한 get 메서드는 모든 문자를 소문자로 변경해서 반환하므로 정보에 접근하는 사용자는 내부 데이터에 어떤 변화가 발생했는지 알 수 없다. 데이터의 포장을 위한 pack 메서드는 두 개의 인자를 받는데, 첫 번째는 언제 누구에게 전달했는지 기록하기 위한 정보이고, 두 번째는 유리 상자의 초기화 조건이다. CopyDetectionGlassBox는 내부적으로 작은 데이터베이스를 가지는데 여기에는 지금까지 pack 메서드를 통해 생성했던 GlassBox들을 기록해둔다. 이 정보를 가지고 후에 유리 상자의 유효성 여부와 유출 데이터 추적을 수행할 수 있다.

〔그림 5〕는 CopyDetectionGlassBox를 이용하는 예제이다. Alice가 Eve에게 개인정보를 전달한다. GlassBox를 얻어서 직렬화를 통해 전달하는 코드를 보여준다(조건 4). 네트워크를 이용하는 것이 일반적이나 코드의 간결함을 위해 임시 파일에 객체를 저장했다가 가져오도록 했다. Eve가 해당 데이터를 몰래 다른 사람에게 넘겨주었을 경우 해당 유리 상자를 Alice가 감정할 수 있으며 감정 결과 Eve에게 언제 전달했는지를 확인할 수 있다. 또 하나의 공격으로 Mallory가 데이터 부분만 복사하여 새로운 유리 상자에 담았다고 가정해보자. 이 유리 상자는 Alice가 만든 것이 아니므로 Alice는 쉽게 이 유리 상자가 가짜임을 판별할 수 있게 된다. 이 예제가 의미하는 것은 제 삼자의 개인 정보를 받았을 때 해당 정보가 정상적인 경로로 나에게 전달되었는지를 확인할 수 있음을 의미한다.

V. 관련 연구

본 논문은 개인정보보호에 대한 분류를 유출 경로를 기준으로 나누고 아직 활발히 논의되고 있지 못하고 있는 개인정보의 축적에 대한 대응 방안을 제안하였다. 본 논문과 같은 목표를 가지고 개인정보보호를 위한 모델을 제안하는 논문은 아직 존재하지 않는다.

본 논문이 제안하는 것과 유사한 목표를 가지고 있는 연구 중 가장 널리 알려진 것은 Vanish 〔13〕이다. Vanish는 정보가 송신자로부터 수신자에게 전달될 때 중간 전달자들이 메시지를 보관하는 것을 차단하고자 하는 것이 목표이다. 송신자는 데이터를 임의의 키로 암호화 한 뒤 키를 나누어(secret sharing〔14〕) DHT (distributed hash table)에 분산시켜 보관하고 키의 위치와 암호문, 그리고 몇 개의 키 조각을 얻어야만 복호화가 가능한지를 수신자에게 전달하는 방식을 취한다. 수신자는 위치 정보에 따라 DHT에서 키 조각을 얻어온 후 조합하여 키를 얻고 복호화 하여 정보를 확인할 수 있다. DHT에서 키 조각이 영원히 존재하는 것이 아니라 일정 시간 머물다가 사라지므로 자연스럽게 중간자들이 데이터를 복호화 할 수 없도록 만든다. 데이터를 암호화해서 저장한 뒤 키를 어느 순간 폐기하면 자연스럽게 데이터를 폐기할 수 있다는 아이디어를 이용했다〔12〕. Vanish는 중간자가 데이터를 보관함으로써 데이터가 유출될 수 있는 문제에 접근한 것으로 본 논문과는 목표가 다르다. 본 논문의 유리 상자 모델은 수신자가 데이터를 폐기하지 않고 필요 이상 오랜 기관 보관함으로써 유출 가능성을 가지는 문제에 접근하였다. 수신자는 데이터를 정상적으로 복호화할 수 있는 권한을 가지기

```
import java.security.AccessControlException;
import java.security.Guard;
import java.security.GuardedObject;

public class WithGuardObject {
    public static void main(String[] args) {
        GuardedObject go = new GuardedObject(
                new String("private data"), new GlassBoxGuard(2));
        System.out.println(go.getObject());
        System.out.println(go.getObject());
        System.out.println(go.getObject());
    }
}

class GlassBoxGuard implements Guard {
    private int used;
    private int limit;

    public GlassBoxGuard(int limit) {
        used = 0;
        this.limit = limit;
    }

    @Override
    public void checkGuard(Object object) throws SecurityException {
        if (++used > limit)
            throw new AccessControlException("The bomb exploded.");
        return;
    }
}

결과:
private data
private data
Exception in thread "main" java.security.AccessControlException: The bomb exploded.
        at kr.ac.ajou.dv.glassbox.GlassBoxGuard.checkGuard(WithGuardObject.java:27)
        at ...
```

〔그림 6〕 GuardedObject를 이용한 유리 상자 모델의 구현 예제

때문에 Vanish와 달리 암호화로 문제를 해결할 수 없었다. 따라서 유리 상자 모델에서는 개인 정보의 수신자가 데이터를 안전하게 다루기 위한 방안을 제시하였다. 또한 수신자가 악의적으로 데이터를 복제하여 사용하려 하는 경우 감지하는 방법에 대해서도 제안하였다.

자바 프레임워크에는 객체를 포장하여 수신자(consumer)가 해당 객체에 대한 접근 권한을 가지고 있을 때만 객체를 얻을 수 있도록 하는 Guarded object를 제공하고 있다〔15〕〔16〕. 객체는 데이터를 표현하기 위한 추상적 자료형(abstract data type)으로써의 역할을 수행할 수 있으므로 Guarded object를 이용하여 데이터를 보호할 수 있다. 따라서 유리 상자 모델을 자바 가상 기계에서 동작시키는 경우에는 GuardedObject 클래스를 활용하여 유리 상자를 구현할 수 있다. GuardedObject는 Serializable 인터페이스를 구현하고 있어서 객체의 전달(조건 4)에도 문제가 없다. 〔그림 6〕은 GuardedObject를 이용하여 〔그림 2〕와 같은 간단한 유리 상자를 구현해본 예제이다.

VI. 결 론

본 논문은 개인 정보가 유출될 수 있는 경로를 분류하고 각각에 대한 현재의 대응 기술을 살펴보았다. 분류는 네 가지로, 1) 정보 자체의 노출, 2) 정보 교환 주체의 노출, 3) 필요 이상의 추가 정보 전달, 4) 정보의 남용에 의한 개인 정보 유출이다. 앞의 세 가지 유출 경로에 대한 대응은 각각 암호 기술, 익명 통신 기술, 필요한 정보만 전달하는 프로토콜의 설계로 해결이 가능하다. 그러나 개인 정보가 필요에 따라 사용된 후 폐기되지 않고 남아서 남용될 수 있는 문제는 해결 방안이 존재하지 않았다. 본 논문에서는 개인 정보의 남용 문제를 해결하기 위해 유리 상자 모델을 제안하였으며 모델의 구현 방안을 제시하였다. 개인정보를 다루는 시스템은 유리 상자 모델을 사용하여 개인 정보를 항상 유리 상자에 담아 보관하고 이용함으로써 안전하게 데이터를 사용할 수 있다. 또한, 악의적인 사용자가 데이터의 자가 초기화를 피하기 위해 복제를 시도하고 이를 유통하는 경우에 대비하여 복제 감지 기법을 함께 제안하였다.

참고문헌

〔1〕 R. Cardoso and V. Issarny, "Architecting pervasive computing systems for privacy: A survey," In Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture(WICSA '07), pp. 26, Jan. 2007.

〔2〕 S. Bellovin, "Security and Privacy: Enemies or Allies?," Security & Privacy, IEEE, vol. 3, no. 3, pp. 92, June 2005.

〔3〕 D.L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, vol. 24, no. 2, pp. 84-90. Feb. 1981.

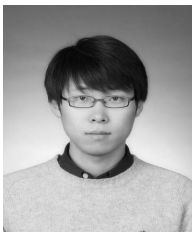
〔4〕 R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," In Proceedings of the 13th conference on USENIX Security Symposium(SSYM'04), vol. 13. pp. 21, June-July 2004.

〔5〕 A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management v0.34,", http://dud.inf.tu-dresden.de/

literatur/Anon_Terminology_v0.34.pdf, 2010년 8월 10일
〔6〕 W3C, "Platform for Privacy Preferences (P3P) Project," http://www.w3.org/P3P, 2013년 7월 25일
〔7〕 K.G. Shin, X. Ju, Z. Chen, and X. Hu, "Privacy protection for users of location-based services," Wireless Communications, IEEE, vol. 19, no. 1, pp. 30-39, Feb. 2012.
〔8〕 F. Stajano and R. Anderson, "The grenade timer: Fortifying the watchdog timer against malicious mobile code," In Proceedings of 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000), pp. 1-5, Oct. 2000.
〔9〕 B. Blakley, "The emperor's old armor," In Proceedings of the 1996 workshop on New security paradigms(NSPW '96), pp. 2-16, Sept. 1996.
〔10〕 Sun Microsystems, Inc. "The Java Language Specification," 3rd Ed., http://docs.oracle.com/javase/specs/jls/se5.0/html/j3TOC.html, 2013년 7월 16일
〔11〕 K.D. Mitnick and W.L. Simon, The Art of Deception: Controlling the Human Element of Security, John Wiley & Sons, Inc., 2003.
〔12〕 D. Boneh and R.J. Lipton, "A revocable backup system," In Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography(SSYM'96), vol. 6. pp. 9. 1996.
〔13〕 R. Geambasu, T. Kohno, A.A. Levy, and H.M. Levy, "Vanish: increasing data privacy with self-destructing data," In Proceedings of the 18th conference on USENIX security symposium(SSYM'09), pp. 299-316, Aug. 2009.
〔14〕 A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612-613, Nov. 1979.
〔15〕 L. Gong and R. Schemers, "Signing, Sealing, and Guarding Java Objects," In Mobile Agents and Security, Springer-Verlag, UK, pp. 206-216, Aug. 1998.
〔16〕 S. Oaks, Java Security, 2nd Ed., O'Reilly Media, Inc., Jun. 2013.

〈저자소개〉

김 종 욱 (Jonguk Kim) 학생회원
2004년 2월: 아주대학교 정보및컴퓨터공학부 학사
2006년 8월: 아주대학교 정보통신전문대학원 석사
2006년 9월~현재: 아주대학교 정보통신전문대학원 박사과정
〈관심분야〉 키 관리, 익명 통신, 인증, 개인정보보호



강 석 인 (Sukin Kang) 학생회원
2008년 2월: 아주대학교 정보및컴퓨터공학부 학사
2008년 2월~현재: 아주대학교 대학원 컴퓨터공학전공 석박사통합과정
〈관심분야〉 네트워크 보안, 인증, 키 관리

홍 만 표 (Manpyo Hong) 종신회원
1981년: 서울대학교 계산통계학 학사
1983년: 서울대학교 계산통계학 석사
1991년: 서울대학교 병렬처리 전공 박사
1983년~1985년: 울산공과대학 전임강사
1985년~현재: 아주대학교 교수
〈관심분야〉 정보보호, 악성코드, DDoS, 스마트그리드 보안, 클라우드 보안