情 報 保 護 學 會 誌
第21卷 第4號, 2011. 6

VRM 관련 오픈소스 프로젝트 동향

김 승 현*, 김 석 현*, 진 승 헌**

요 약

개인화 기술은 사용자의 취향과 의도를 분석하여 많은 정보와 서비스 중에서 사용자에게 필요한 부분을 선별해준다. 이를 위해서는 사용자의 개인정보가 요구되는데, 지금까지의 기업 위주로 개인정보를 관리하는 방식은 다양한 문제를 야기했다. VRM은 사용자가 본인의 정보를 직접 관리하는 기술로, 필요에 따라 기업이나 친구 같은 제3자와 개인정보를 공유할 수 있는 기술이다. 본 논문에서는 VRM 기술을 소개하고, 관련된 오픈소스 프로젝트 동향을 정리 분석하였다.

Ⅰ. 서 론

기술이 발전함에 따라 기업이 제공하는 서비스의 품질 또한 높아지고 있다. 하지만 너무 많은 정보와 기능 때문에 오히려 사용자의 혼란을 초래하는 경우가 많다. 사용자의 취향이나 의도를 파악하지 못한 정보는 스팸처럼 쓸모없는 것으로 취급되며, 유용한 정보임에도 불구하고 사용자가 찾기 어려워서 제대로 활용하지 못하기도 한다. 그래서 기업은 사용자의 개인정보를 최대한 수집하여 서비스를 구상하고, 사용자의 취향을 예측하여 개인화된 서비스를 제공한다.

개인화는 기업과 사용자 모두에게 이익이 된다. 그러나 사용자의 개인정보가 기업에 의해 수집 관리되면서 사용자는 이를 알지 못하기 때문에 많은 문제가 제기된다. 1) 기업이 관리하는 개인정보의 사후관리 부재로 인해 프라이버시 침해 사고가 빈번하다. 2) 기업은 개인정보를 수집 및 활용하기 위해 CRM(Customer Relationship Mangement)[1]을 구축하는데, 막대한 유지비용이 소요된다. 또한 3) 최근의 개인정보보호법[2]과 같은 정부의 규제를 준수하기 위해서는 많은 비용과 노력이 추가로 요구된다. 4) 개별 기업이 수집 가능한 개인정보는 일부에 불과하고 변화하는 사용자의 정보를 추적하지 못하기 때문에 효과적인 개인화 서비스를 제공하기 어렵다.

이 때문에 사용자가 직접 본인의 개인정보를 쉽게 열람하고 제어(변경, 파기, 고지)할 수 있을 뿐만 아니라, 기업이 사용자로부터 실시간으로 개인정보를 공유받을 수 있는 수단이 필요하다. VRM은 사용자가 개인의 정보를 관리하는 기술로, 필요에 따라 기업이나 친구 같은 제3자와 해당 내용을 공유할 수 있는 기술이다. 본 논문에서는 VRM이라는 기술을 소개하고, 관련된 오픈소스 프로젝트 동향을 정리하였다.

본 논문은 다음과 같이 구성된다. 2장은 VRM 기술을 소개하고, 3장은 VRM과 관련된 5개의 오픈소스 프로젝트를 자세하게 정리한다. 4장에서 VRM 기술 및 현황을 고찰하고, 5장에서 결론을 맺는다.

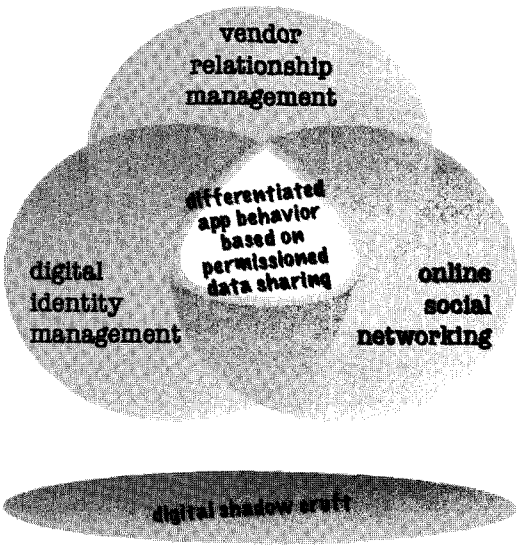
Ⅱ. VRM(Vender Relationship Management)

VRM 개념은 하버드 대학의 Berkman Center for Internet and Society에서 수행한 ProjectVRM[4] 프로젝트에서 시작되었다. VRM은 기업이 보관하는 고객 데이터를 고객이 직접 관리한다는 아이디어이다. VRM의 철학은 개인정보 보호에 중점을 두고 있으며, ID 관리 기술과 흐름을 같이 하기 때문에 사용자 중심으로 명시적인 승인을 통한 안전한 정보 공유를 지향한다[3].

개인정보의 공유 채널로는 전통적인 이메일이나 웹(HTTP, RSS 등)뿐만 아니라 최근 급격히 성장하는 소셜 네트워크(Facebook, Twitter 등)도 포함된다. 특히 소셜 네트워크의 경우, 사용자들의 개인정보가 빈번히

* 한국전자통신연구원 지식정보보안연구부 (ayo@etri.re.kr, ksh4uu@etri.re.kr)
** 한국전자통신연구원 지식정보보안연구부/팀장 (jinsh@etri.re.kr)

공유되지만 프라이버시 문제에 대해서는 미흡한 대응을 보이고 있다. 따라서 소셜 네트워크 상에서 사용자와 사용자간, 사용자와 기업 간에 안전하게 정보를 공유하고, 이를 통해 비즈니스를 창출할 수 있는 보안 채널을 구성하는 것이 VRM의 역할이다. [그림 1]은 VRM 기술이 ID 관리 기술과 소셜 네트워크 사이에서, 고객의 승인에 기반한 데이터 공유를 통해 차별화 시킬 수 있는 부분을 보인다[5].



[그림 1] VRM, ID관리기술, 소셜네트워크 간의 관계

VRM은 고객의 관점에서 개인정보를 기업에게 제공하기 위한 다양한 기능을 제공하는데, 그 중에서 주목할 만한 4가지 기능을 소개한다[6].

2.1 개인용 데이터 저장소(Personal Data Store)

PDS는 고객의 개인정보와 온·오프라인 활동 기록을 로컬에 보관하고, 필요에 따라 선택적으로 활용하는 저장소이다. 기업에서 관리하는 CRM 정보와 유사하지만, PDS의 경우 실시간으로 변화하는 고객의 민감한 데이터라는 점과 특정 기업에 국한되지 않는 해당 고객 자체의 정보라는 차별성이 있다. CRM에서는 PDS처럼 민감하고 정확도 높은 자료를 확보하는 것은 불가능에 가깝다. PDS로 수집한 데이터를 바탕으로, 고객은 VRM이 제공하는 다른 기능(개인용 데이터 분석, 제안요청서, 역 메시징 등)들을 활용할 수 있게 된다.

2.2 개인용 데이터 분석(Personal Data Analytics)

PDS에서 수집된 정보를 분석하면, 고객의 다양한 패턴을 측정할 수 있다. 개인의 소비성향(금융정보), 칼로리 섭취 현황(식습관), 마일리지 현황(구매 패턴) 등이 그 예이다. 데이터 분석을 통해 고객이 리소스를 잘 관리할 수 있는 부분을 표시하고, 비슷한 성향의 타 고객과 비교하여 고객의 패턴을 개선하는 것도 가능하다. PDS에서 특정 일정이 계획될 경우, 타 고객들의 변화된 패턴을 통해 고객의 패턴을 예측할 수 있다. 이에 따라 일상의 변화(결혼, 출산, 이혼, 여행, 이사 등)로 인해 고객의 패턴이 급격히 흐트러질 때에도 균형 있게 조절하는 방안을 제시한다.

2.3 제안 요청서(Request for Proposal)

RFP는 고객이 기업에게 보내는 제안서를 의미한다. 제안서 내에는 고객이 제공받고 싶은 제품 또는 서비스에 대한 요구사항이 포함되어 있으며, 제안서는 하나 이상의 기업에게 전송된다. 기업은 고객의 제안서가 명세한 요구사항을 검토한 뒤, 자신들의 제안/가격/약관을 제시한다. 고객은 기업의 조건을 검토하여, 최종 구매를 판단한다.

이 기능은 '역경매'와 동일하다고 볼 수 있지만, VRM에서는 역경매 기능을 제공하는 특정 플랫폼에 국한되지 않는다. 고객은 기업의 특정 창구에 직접 제안서를 보내거나, 브로커(intermediary)를 통해 대행할 수 있다. 기업의 응답 또한 개인에게 직접 전달되거나 브로커를 통해 그룹 단위로 전달할 수 있다.

또한 VRM은 익명으로 거래를 진행하는 기능을 제공한다. 브로커는 특정 기업이나 여러 기업을 선정하여 특정수의 고객들이 특정 제품이나 서비스를 찾고 있다고 알려주지만, 기업들은 트랜잭션 도중에는 브로커를 통해서만 일부 고객 정보에 접근할 수 있다. 물론 최종 거래 단계에서는 고객이 직접 기업에 접근하여 브로커를 배제하고 민감한 정보를 주고받을 수도 있다.

2.4 권한 관리와 역 메시징

고객은 PDS의 특정 데이터를 어떤 기업에게 공개할지를 제어할 수 있다. 거래를 위해 한 번만 제공하거나, 거래 없이도 변경 내역을 기업에게 제공하는 채널(역

메시징)을 맺을 수 있다. 이 과정에서 고객은 어떤 목적으로 어떤 약관과 조건 하에서 개인정보가 활용될지를 선택할 수 있다.

Ⅲ. VRM 관련 오픈소스 프로젝트

본 장에서는 VRM과 관련된 여러 작업 중에서, 오픈소스로 진행되는 5개 프로젝트를 선별하여 상세하게 조사하였다. 조사된 프로젝트는 Higgins, Danube, Locker, Mine!, Nori이다.

3.1 Higgins

Higgins[7]는 이클립스 재단에 의해 수행되는 오픈소스 인터넷 ID 관리 프레임워크로 다양한 기기, 웹사이트, 응용프로그램에 산재된 ID, 프로파일, 소셜관계 정보를 통합하여 관리한다. Eclipse 라이선스를 준용하며 Higgins 1.0은 2008년 2월에 종료되었고, 2010년부터 2012년 6월까지 2.0 버전을 개발 중이다.

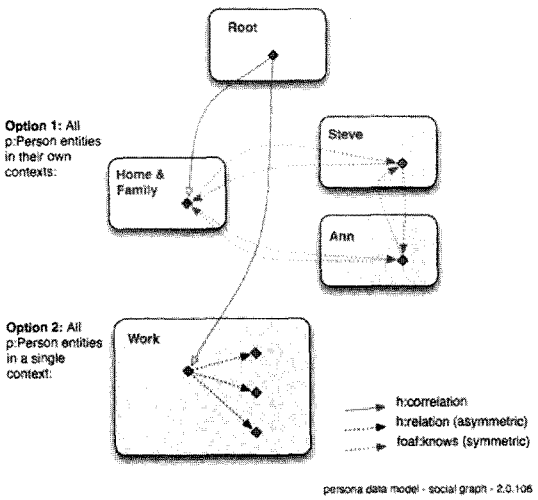
Higgin 프로젝트의 목적은 정보를 수집하고 관리, 공유하는 PDS(Personal Data Service)를 개발하는 것이다. PDS는 3가지 기능을 요구한다. 첫 번째는 데이터 모델이 정규화되어 있어서 데이터를 조회하거나 수정, 갱신이 쉽게 이루어질 수 있어야 한다. 두 번째는 검색(Discovery) 기능으로 다른 사람/조직/어플리케이션을 검색하고 정보를 교환할 수 있도록 검색 API를 지원해야 한다. 세 번째는 상호호환성(Interoperability)이다. 각 PDS는 다른 기관의 네트워크에서 관리되더라도 데이터를 상호 교환할 수 있어야 한다. 개인이 직접 호스팅하는 PDS는 보통 다른 PDS에 저장된 객체의 링크를 보관하는데, 이 링크는 소셜 그래프(social graph)[8] 형식을 가진다.

3.1.1 데이터 모델

PDS에서 사용자에 의해 생성, 저장 및 공유되는 데이터는 다양성을 지원하는(rich) 공통(common) 데이터 모델로 매핑된다. Higgins 프로젝트에서는 사람과 소셜 네트워크를 표현하기 위해 개발된 공통 데이터 모델을 Persona Data Model 2.0[9]이라고 부른다. 데이터 저장소의 핵심은 컨텍스트(context)라고 불리는 컨테이너 집합이다. 각 컨텍스트는 사용자의 디지털 ID의 일부인 페르소나(persona)를 표현한다. 각 페르소나 인스턴스는 여러 속성들을 저장한다.

컨텍스트는 사용자에게 카드 메타포로 보여진다. 예를 들어 운전면허증, 집주소, 명함, 신용카드 정보 등의 페르소나가 하나의 카드가 된다. 컨텍스트는 사용자의 친구나 동료를 표현할 수도 있다. 외부의 서비스를 이용하기 위해, 사용자는 여러 개의 카드를 조합하거나 선별하여 사용한다. 온라인 쇼핑에서 주소, 카드정보, 성인여부를 전달해야 할 때, 집주소 카드, 신용카드, 운전면허증의 페르소나를 선별하여 최소화된 정보만을 사용할 수 있다.

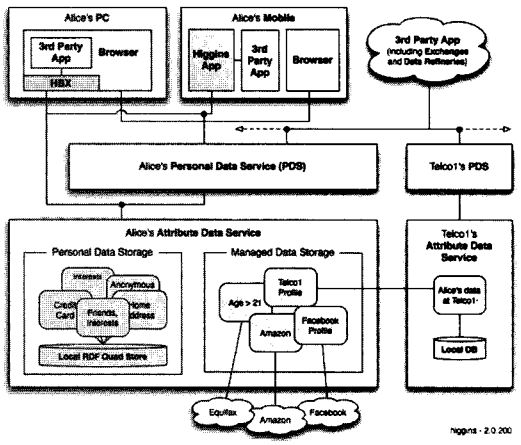
[그림 2]는 컨텍스트를 표현한 person graph를 보인다[10]. 최상단의 root는 해당 사용자를 나타내며, root와 연결된 둥근 모서리 사각형은 P:Person 노드로 각기 다른 페르소나를 나타낸다. 컨텍스트는 XML 형식을 가지며, 간단한 정보뿐만 아니라 복잡한 구조도 표현 가능하다. H:correlation은 사용자의 페르소나들 간의 관계, H:relation은 사용자 간의 단방향 관계, foaf:know는 사용자 간의 양방향 관계를 나타낸다.



[그림 2] Higgins 프로젝트에서 Context를 표현한 Person Graph

3.1.2 아키텍처

[그림 3]은 Higgins 프로젝트의 PDS 아키텍처를 보인다[11]. 주 컴포넌트로는 PDS, ADS, 서드파티 앱, HBX/AC가 있다.



〔그림 3〕 Higgins의 PDS 아키텍처

PDS(Personal Data Service)는 다양한 출처(소셜네트워크, 이동통신사, 병원 등)로부터 가상으로 정보를 통합하고 사용자가 관리할 수 있도록 보여준다. PDS는 웹브라우저, 어플리케이션, 모바일 단말에 구축될 수도 있다. 사용자는 PDS에서 자신의 정보를 제어하고, 선택된 사람 또는 조직에게 일부를 공유한다. 서드파티로부터 검증받은 정보를 공유하거나, 런타임 환경에서 구동하는 APP을 자체 내장하거나, two factor 인증 및 로컬 저장소의 정보를 암호화 관리하는 기능 등을 Higgins PDS에서 활용할 수 있다.

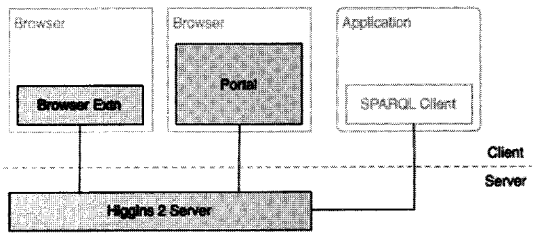
ADS(Attribute Data Service)는 데이터 저장소의 추상화 계층으로 공통 데이터 모델과 매핑된다. 데이터 저장소는 personal/managed로 구분된다. Personal은 로컬에 저장되는 페르소나들을 관리하며, 외부 서비스나 친구들에게 페르소나를 선택적으로 링크한다. 또한 데이터의 암호화, 백업, 동기화 등이 기능이 제공된다. Managed는 컨텍스트 컨테이너로 표현되는 외부의 데이터 서비스로, 하나 이상의 페르소나를 원격에 보유/관리한다. 예를 들어, 페이스북의 사용자 프로파일은 페이스북 컨텍스트에서 하나의 페르소나로 대변된다. 사용자의 친구 또한 사용자 페르소나와 연결된 다른 페르소나 객체가 된다.

서드파티 앱은 데이터 교환과 정련(refinery)를 수행하는 PDS 연동 프로그램이다. 데이터 정련은 PDS의 데이터를 읽고 가공한 뒤에 PDS 사용자에게 돌려주는 분석(analytics), 추론(inferencing), 분절(segmentation) 등의 단계를 거친다. 이 정련 단계를 통해 덜 식별되었던 로우(raw) 데이터의 가치를 높인다.

AC(Active Client)/HBX(Higgins Browser eXtention)는 웹 기반의 플러그인으로, 수동적인 웹브라우저를 액티브 클라이언트(active client)로 만들어주는 네 가지 부가 기능을 제공한다. 1) 사용자가 웹 브라우징하는 동안의 정보를 수집한다. 2) web 증강화(augmentation)로 웹 브라우저에 컨텍스트 특화된 정보를 덧씌우거나 자동화된 폼 필링을 통해 사용자의 웹 경험을 개선한다. 3) 악의적인 웹사이트에 대한 피싱 방지를 지원한다. 4) 개인 데이터를 암호화하여 클라우드 기반의 PDS에 데이터를 노출시키지 않는다.

3.1.3 구현

현재까지 데모용으로 개발된 Higgins 프로젝트는 서버, 클라이언트(포털, 브라우저 확장)를 개념적으로 구현했다. 서버는 Persona Data Model 2.0을 준용하여 사용자 데이터를 RDF[12] 형식으로 저장했다. 브라우저 확장과 포털은 자바스크립트를 활용하여 데이터를 로드하여 사용한다. [그림 4]는 데모 시스템의 간략한 구조를 보인다[13].



〔그림 4〕 Higgins의 데모 아키텍처

3.2 Danube

Danube[14]는 인터넷 환경에서 ID 관리 도구와 PDS를 제공하는 오픈소스 프로젝트로, TBD 라이선스를 준용한다. 프로젝트의 핵심은 XDI 기반의 개인용 데이터 저장소인데, 이 저장소는 사용자의 제어 하에 개인 데이터를 관리하는 데이터베이스이다. 이 데이터베이스를 바탕으로, 연방화된 소셜 웹(Federated Social Web)[15] 같이 조직과 개인정보를 선택적으로 공유하는 어플리케이션을 개발할 수 있게 된다.

Danube 프로젝트의 장기적인 목적은 상호호환성

(interoperability)과 데이터 이동성(data portability)이다. 기술적 측면에서 상호호환성은 제공자가 어떤 코드를 사용하는지에 무관하게 동작 가능해야 하며, 이를 위해 XDI[18], RDF[12], Ostatus[16] 같은 공개 프로토콜이 필요하다. 그리고 데이터 이동성의 경우, XDI와 같은 표준 규격을 활용하기 때문에 다른 어플리케이션에서의 데이터 사용 및 다른 포맷으로의 전환이 용이하다.

3.2.1 기술

Danube PDS의 핵심 기술은 XRI[17]와 XDI[18]이다. XRI(eXtensible Resource Identifier)는 개인 또는 조직의 식별자로 사용되는 문법 및 해석(resolution) 기술로, I-Names와 I-Numbers 쌍으로 구성된다. I-Name은 사람이 읽을 수 있는, 사용자의 PDS 계정에 대한 완전히 휴대 가능한(fully portable) 식별자이다. I-Number는 재할당되지 않는 식별자로 스위스 은행 계좌 번호처럼 비밀리에 사용자 PDS를 보호한다. XDI(XRI Data Interchange)는 XRI에 기반한 데이터 모델로, 인터넷을 통한 사용자의 PDS와 조직/개인 간의 데이터 공유, 연결(linking), 동기화 작업에 사용된다. XDI는 Link Contract라고 불리는 모델을 통해 접근성, 그룹 관리, 접근 제어를 제공한다.

관련 프로젝트와 비교할 때, Danube 프로젝트는 상대적으로 보안 기능에도 초점을 맞추고 있다. PDS 내의 개인정보는 모두 암호화 되어 관리한다. 접근제어와 모니터링을 제대로 수행하기 위해, Danube는 하나의 컴포넌트만 실제 데이터(pds-core)에 접근 가능하다. 모든 컴포넌트와 프로토콜 엔드포인트는 접근 제어와 모니터링 대상이며, 실제 데이터와 동일한 아키텍처 수준으로 개발된다.

Danube PDS는 공개키 기반구조를 사용한다. 모든 I-Name은 비밀키/공개키 쌍으로 임의의 데이터에 대한 서명 및 검증을 수행한다. 임의의 데이터로는 개인 간에 주고받는 사적인 메시지에서부터 조직과 맺는 Link Contract가 해당한다. 공개키 기반구조를 채택하였기 때문에 Danube PDS는 프로바이더, 개인, 조직 간의 분산 시스템 구조에서도 책임(accountability), 보안, 신뢰를 보장할 수 있다.

3.2.2 데이터 모델

Danube에서 XRI/XDI로 표현하는 데이터 구조로는 Profile, Relationship, Group, Link Contract가 있다. Profile은 사용자가 직접 생성하고 제어하는 정보로, 여러 개의 페르소나로 이루어질 수 있다. Profile은 사용자의 접근 제어(프라이버시)내역에 따라 공개여부가 결정되는데, 외부 프로그램과의 연동을 위해 FOAF[19], hCard[20], Portable Contacts[21] 등의 공개 스펙을 활용하기도 한다.

Relationship은 사용자가 누구와 데이터를 공유할지, 어떤 어플리케이션을 사용할지를 정의한다. Danube는 Relationship을 개별(Individual), 조직 (Organization)으로 분류한다. 개별 Relationship은 매우 다양한 형식을 가지며, 친구들 간의 데이터를 공유하기 위한 목적으로 활용된다. 반면에 조직 Relationship은 기업과 사용자 간의 데이터 공유가 목적이며 크게 3가지 타입의 연산(한번 읽기, 쓰기, 읽고 쓰기)을 제공한다.

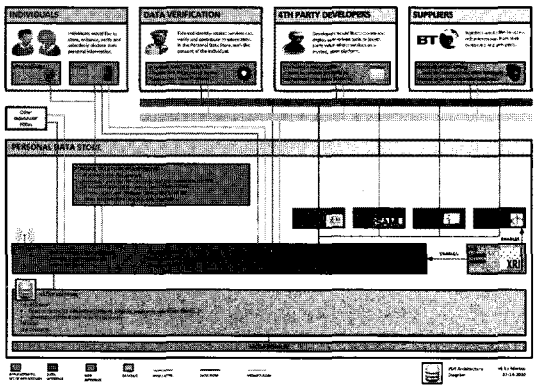
Group은 가장 높은 수준의 객체로서, 개별 PDS 계정과 같은 의미를 가진다. Group은 다른 데이터 모델과 완전히 독립적이고 분산되어 있다. 따라서 자체적으로 데이터, 접근 제어 방식, 관련된 어플리케이션의 목록을 보유한다.

마지막으로 Link Contracts는 XDI의 기본 접근제어 메커니즘으로, 엔티티간의 관계를 수립하기 전에 필수적으로 요구되는 요소이다. Link Contract는 PDS의 실제 데이터와 매칭되어 존재한다. Link Contract 데이터는 그 자체로 완전히 독립적이고, 아키텍처 구조의 맨 아래에서 구동하여 최대한의 보안을 제공한다. 어떤 관계나 어플리케이션이든지 실제 데이터에 접근할 때는 Link Contract에 명시된 규칙과 용어 하에서만 가능하다. Link Contract는 보안을 위해서 특정한 인증 기술을 강제하기보다는, 서로가 합의한 인증 방식을 명시 할 수 있다. 또한 Link Contract는 법적인 요소와 인간이 읽을 수 있는 문구를 포함한다.

3.2.3 아키텍처

[그림 5]는 Danube 아키텍처의 컴포넌트 구조를 보인다[22]. Danube의 컴포넌트는 모듈화 되어있기 때문에 핵심 PDS 서비스(pds-core)만으로 가능한 시나리오에서부터 hCard (pds-endpoint-hcard), OAuth 2.0[23] (pds-endpoint-oauth2), Web 인터페이스(pds-web)를 고려한 복잡한 시나리오까지 자유롭게 구성 가능하다. 또한 기존 레거시 시스템 환경을 고려하여 간단한 설정

만으로도 사용자에게 PDS 계정을 제공할 수 있다.



〔그림 5〕 Danube 아키텍처

3.2.4 구현

Danube 아키텍처의 구현을 위해서 다른 오픈소스들이 활용되었다. Danube를 구성하는 핵심 기술인 XRI, XDI를 위하여 OpenXRI (Apache 2.0 라이선스)[24]와 XDI4j (Eclipse 라이선스)[25] 라이브러리를 채택하였다. Danube 아키텍처는 2010년 9월에 코어 XDI 기능을 개발하는 0.1 버전을 시작하여 2010년 12월에 0.3 버전까지 개발하였다. 0.3 버전에서는 경량화된 XDI Graph Database, XDI Messaging Engine, XDI Endpoint가 포함되었다.

3.3 Locker

Locker[26]는 산재되어 있는 개인 데이터를 모두 수집하여 관리하기 위한 오픈소스 프로젝트로 BSD 라이선스를 준용한다. Locker 프로젝트는 단독으로 진행하지 않고 TeleHash, Singly 같은 다른 프로젝트와 협업한다. TeleHash[27]는 P2P 통신 프로토콜이고, Singly[28]는 Locker 프로젝트에 신뢰 보안 계층을 제공하는 기술을 개발한다. Singly의 경우, 상세한 자료는 공개되지 않고 있다.

3.3.1 기술

TeleHash[27]는 P2P 방식으로 다른 Locker와 연결하여 사용자 데이터를 공유하는 채널을 제공하는 프로토콜로 Public Domain 라이선스를 준용한다. TeleHash는 완전히 분산화된 방식으로 JSON[29] 데이터를 실시간으로 교환한다. 특정 어플리케이션이나 컨텐츠 구조에 의존하지 않은 형식이며, 서버나 중앙의 제어를 요구하지 않는 분산화 구조이다. TeleHash는 분산 해쉬 테이블로 많이 사용되는 Kademlia[30]로 라우팅 시스템을 구축하여, UDP 채널로 간단한 JSON 객체를 주고받는 식으로 동작한다. TeleHash에서 라우팅되는 데이터는 SHA 해쉬값을 사용하는데, 이는 특정 어플리케이션에 특화된 정보 또는 URL처럼 일반적인 정보이다.

TeleHash의 모든 엔드포인트는 Switch라고 불리는데, 한 Switch에는 하나의 어플리케이션이 매칭된다. Switch는 ID:PORT 구조로 식별할 수 있으며, DHT에서 UDP 패킷으로 주고받는 데이터는 IP:PORT를 SHA1 해쉬한 값이다. Switch는 네트워크에 신호를 보내거나, 특정 신호를 수신하는 두 가지 역할을 수행한다. 하지만 다른 Switch와 연결하여 어플리케이션에 특화된 부가적인 작업을 수행할 수도 있다.

3.3.2 아키텍처

API와 FEED[35]를 활용하여 Locker 프로젝트는 여러 웹서비스의 데이터를 수집한다. 사용자는 데이터를 본인의 서버에 저장하거나 블로그 플랫폼인 WordPress[31] 같은 호스팅 서비스에 저장할 수 있다.

아키텍처는 크게 Connectors, Collections, Apps 라는 3가지 컴포넌트로 구성된다. Connectors는 사용자의 데이터가 있는 장소에 어떻게 접속하고 연동하는지를 알고 있는 컴포넌트이다. 여기서의 장소란, 외부 사이트 계정이나 서비스, 데스크톱 어플리케이션, 휴대폰, 또는 임의의 디바이스까지를 가리킨다. Collections는 여러 장소에 저장된 사용자 데이터를 구조화하여 범용 집합으로 만드는 컴포넌트로, 장소, 연락처, 메시지, 상태정보, 링크 등의 데이터를 취급한다. Apps는 Locker에 저장된 사용자 데이터를 유용하게 가공하는 컴포넌트이다. 사용자는 특정 App이 어디에 저장된 사용자 정보를 사용할지와 온라인 계정에 연결할지 여부 등을 제어할 수 있다.

3.3.3 구현

Locker 프로젝트는 현재 2가지 방식으로 운용 가능

하다. 하나는 소스코드를 다운받아 사용자가 직접 서버를 구동하는 방식이고, 다른 하나는 WordPress[31] 같은 호스팅 서비스를 활용하는 방식이다. 서비스를 일단 구동하면 사용자의 접근 허락을 받은 이후 모든 종류의 데이터를 개인 Locker에 수집 관리한다. 트위터, 사진, 비디오, 웹사이트 방문 기록, 포스퀘어(FourSquare) 체크인, 심장 모니터와 같은 실생활에서의 센서 데이터, 건강 기록이나 금융 거래 같은 트랜잭션 기록 등이 수집 대상이다.

사용자 네트워크에서의 검색 및 공유 기능은 Tele-Hash의 P2P 기능을 활용한다. Switch는 현재 perl, c, ruby, erlang으로 구현되어있다[32].

3.4 MINE!

Mine! 프로젝트[33]는 사용자의 다양한 데이터(컨텐츠, 관계, 트랜잭션, 지식(knowedge))를 수집, 배열(분석, 조작, 결합, 매쉬업)하여 사용자의 요구사항과 취향을 도출하고, 웹에 연결된 상태에서 적절한 포맷으로 이 데이터를 공유하는 서비스를 제공한다.

Mine! 프로젝트의 목적은 다른 솔루션에 대한 인프라 역할을 수행하는 것으로, VRM, self-defined ID, 인증, 데이터 이동성 등의 기능을 제공한다.

3.4.1 데이터모델

Mine! 프로젝트에서 객체나 문서는 반투명(semi-opaquely)한 상태로 취급되는데, 이는 데이터가 특정 플러그인에 의존되지 않기 위해서이다. 이 때문에 어떤 종류의 데이터도 관리할 수 있다. Mine!에서는 HTTP 통신 채널과 Atom[34] 형식만을 유일하게 강제한다.

Mine!은 FEED[35] 방식을 통해 데이터를 공유한다. 하지만 블로그에서 사용하는 FEED와는 달리, Mine!은 승인된 서드파티에만 FEED를 제공하며, 접근 내역을 관리한다. 또한 의료기록과 같은 중요정보이거나 서드파티가 보증하는 데이터(assertion)일 경우, 암호화, 접근제어와 같은 다른 기술들이 추가로 적용된다.

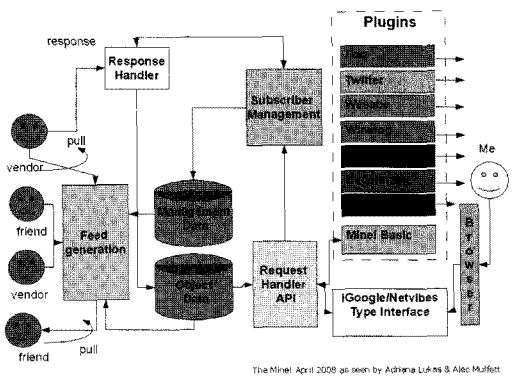
사용자의 데이터를 활용하는 사례 중, 서드파티 데이터를 활용하고 무결성을 보장해야 하는 경우가 있다. 예를 들어 사용자의 신용 등급을 제공해야 할 때, 신용보증회사는 디지털 서명된 방식으로 사용자의 신용 내역을 Mine!에 발급한다. 그러면 사용자는 원할 때마다 이 내역을 다른 기업과 공유한다.

다른 방식으로는 기업이 사용자의 신용 내역을 요구할 때 디지털 서명된 유효기간이 설정된 인증서를 사용자에게 전달하는 것이다. 사용자는 자신의 FEED에 이 인증서를 설정하지만 변조는 할 수 없다. 이 FEED는 신용보증회사의 VRM과 연계되어, 사용자는 실제 데이터를 얻을 필요 없이, 해당 기업만 사용할 수 있는 사용자의 신용 내역이 기업의 인증서로 암호화되어 전달되는 것이다.

3.4.2 아키텍처

[그림 6]은 Mine! 아키텍처를 구성하는 모듈을 보여준다[36]. 왼쪽의 원으로 표시된 것은 외부 엔티티로, 기업이나 친구에 해당한다. 외부 엔티티가 제공하는 사용자의 정보는 Feed Generation 컴포넌트에서 취합하여 Management/Object Data로 저장된다. 반대로 Feed Generation 컴포넌트는 유일무이한 링크 주소로 FEED를 생성하고 이를 친구나 기업에게 제공한다. 기업은 이 링크를 통해 사용자의 정보에 접근하고 응답을 남기거나 제안을 할 수 있다. 링크에 유효기간을 설정하여 자동으로 정보 제공을 중단하거나, 스팸여부 또는 사용자 판단에 의해 직접 링크를 제거할 수도 있다. 사용자는 브라우저 또는 별도의 플러그인 통해 데이터를 설정/조회/관리할 수 있다.

Response Handler 컴포넌트는 사용자 측면에서 트랜잭션을 처리하는 시작점이다. 외부 엔티티가 사용자의 데이터를 요청할 경우 Response Handler 모듈을 경유하며, Request Handler API를 통해 사용자의 설정 정보를 참조하거나 직접 사용자에게 물어봐서 처리한다.



〔그림 6〕 Mine! 아키텍처

3.4.3 구현

Pymine[37] 은 Mine!을 제대로 구현한 첫 번째 소프트웨어로 Apache 2.0 라이선스를 준용한다. Django 1.0 프레임워크[38]에서 개발되어 있으며, Google App Engine[39]으로 포팅될 계획이다. 현재 Mac OS X와 Ubuntu Linux 플랫폼에서 구동 가능하다.

3.5 Nori

Nori 프로젝트[40]는 상호운용하는 PDS를 지향한다. 다양한 PDS 구현물 간에 데이터와 레퍼런스를 공유하기 위한 상호운용 표준화를 제공하는 목적을 가진다.

Nori 프로젝트는 Personal Data Ecosystem 컨소시엄[41]의 일부이며 2010년부터 시작하였다. 이 컨소시엄은 사용자 중심의 데이터와 ID 관리, 프라이버시, 결제 이슈를 해결하는 목표를 가진다.

3.5.1 데이터모델

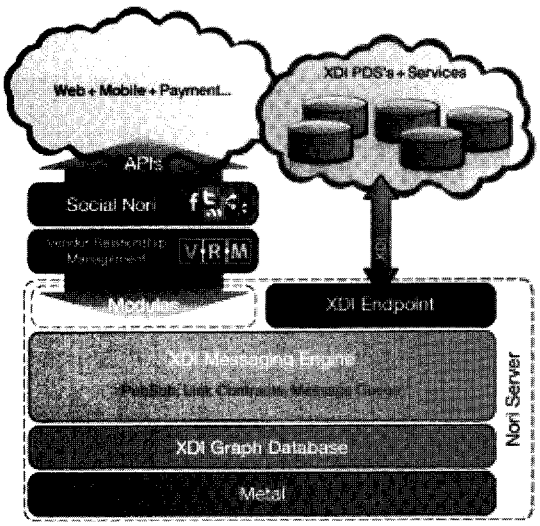
Nori 프로젝트는 XDI[18], RDF[12], OAuth[23] 등의 기존 기술을 활용한다. XDI는 데이터를 주고받는 간단한 방법을 제공하며, XDI Link Contract 포맷으로 데이터 내부에 권한을 포함시킨다. PDS의 데이터는 RDF 형식으로 변환 가능하다. Nori 프로젝트에서는 XDI와 RDF 간의 매핑을 위한 작업을 수행 중이다. 이를 통해 Higgins[7]의 RDF와 Danube[14]의 XDI간의 호환이 가능해진다.

PDS의 핵심 부분은 경량화된 XDI Graph 데이터베이스, XDI Messaging Engine, XDI Endpoint를 포함한다. 핵심 부분은 그 자체로 XDI 기반의 어플리케이션에 탑재되거나, 모듈을 추가하여 사용성을 높일 수 있다.

3.5.2 구현

[그림 7]은 Nori 프로젝트의 아키텍처를 보인다[42]. Nori는 데이터를 저장하는 목적이 아닌, 관리용의 PDS를 개발한다. Nori의 PDS는 문맥적으로 링크된 데이터를 저장하는 특수용도의 데이터베이스이다. 누가, 무엇을, 언제, 어디에 사용자 데이터를 저장했는지를 메타데이터 형식으로 유지하고 이를 효과적으로 관리/공유한다. 실제 데이터는 외부의 PDS 저장소에 위치하며, 이를 활용하기 위해 XDI를 사용한다.

Nori은 Danube[14]와 Higgins[7] 프로젝트를 기반으로 한다. Nori 프로젝트는 Java 버전의 PDS를 구현중이며, 클라이언트 라이브러리는 Python, Ruby, Scala 언어로 개발하고 있다.



〔그림 7〕 Nori 아키텍처

Ⅳ. 고찰

VRM 기술은 ID 관리 분야의 User-Centric과 거의 동일한 개념이다. 하지만 다른 점이 있다면 ID는 기업이 사용자를 식별하기 위한 의도가 많았지만, VRM이 취급하는 개인정보는 범위가 매우 넓고 사용자의 의도가 개입되어야만 한다는 것이다. 또한 기업이 개인정보를 보유하기 때문에 발생하는 다양한 문제들을 해결하기 위해서는, 개인정보를 사용자가 직접 관리해야 한다는 근본적인 개념의 전환이 필요했다.

VRM의 철학은 매우 명쾌하고 대의적이기 때문에 사용자와 관련 업계에 받아들여질 가능성이 높다고 본다. VRM 개념이 소개된 지 5년 밖에 지나지 않았지만 업계에서는 긍정적인 반응을 보이고 있으며 이미 여러 벤처기업과 프로젝트로 진행되고 있다. 본 논문에는 포함되지 않았지만, 벤처기업의 경우 여러 기업들이 대규모 투자를 받아 개발 중이다.

예상과는 달리 오픈소스 프로젝트로도 다수 진행되고 있으며, 인프라 측면에서의 설계와 개발도 많이 이루어졌다. 다양한 언어와 플랫폼으로 개발이 진행되고 있

으며, 실제로 본 논문에서 소개된 PDS를 설치하는 것 뿐만 아니라 개발에 참여하는 것도 용이하다.

VRM 기술의 특성상, 아키텍처와 데이터는 범용적이고 확장 가능해야 한다. 이 때문에 기존의 공개 기술인 XRI, XDI, RDF, OAuth, PKI, Feed를 활용하였다. Nori 프로젝트의 경우는 메타데이터 개념을 차용하여 VRM 프로젝트 간의 상호호환을 목표하고 있다. 향후에도 데이터 포맷, 데이터 표현 언어, 통신 프로토콜을 상호호환하거나 표준화 할 것으로 예상된다.

하지만 아직 실제로 활용되기에는 미흡한 부분이 보인다. VRM 프로젝트들이 개념 정립 수준으로 구현된 상태이며, 실제로 사용자에게 활용되기 위해서는 개인정보 수집 방식을 다변화하고 사용자 편의성을 개선해야 한다. 또한 수집된 개인정보를 제3자에게 제공하기 전에 가공하거나 프라이버시 등 보안을 고려하는 절차가 더욱 강화되어야 한다. 데이터 모델, 통신 계층 위에서 신뢰 계층에 대한 고려도 필요하다. XDI 등의 기술도 실제 활용과 연계되어 계속 표준화가 진행되고 있기 때문에 꾸준한 개선이 요구된다.

V. 결 론

본 논문에서는 기존에 기업 위주로 수행되었던 개인정보 활용에 대응되는 개념인 VRM 기술을 소개하였다. 또한 VRM 기술을 오픈소스로 구현한 Higgins, Danube, Locker, Mine!, Nori의 5개 프로젝트를 자세히 설명하였다. 상호호환과 확장성을 고려하여 기존의 공개 기술을 활용해 개발이 이루어졌으며, 쉽게 개발에 참여할 수 있다. 하지만 실제로 활용되기 위해서는 개선이 시급한 상황이다.

향후 연구로는 기업들이 추진하는 VRM 기술 동향, VRM과 관련된 여러 컨소시엄의 동향을 조사 분석할 예정이다.

참고문헌

[1] Customer Relationship Management, http://en.wikipedia.org/wiki/Customer_relationship_management

[2] 법제처, 개인정보 보호법, 법률 제10465호, 2011.3.29

[3] 김승현, 진승헌, "VRM: 사용자 중심의 고객관계관리(CRM)", 주간기술동향, pp. 1-12, 2011년. 1월.

[4] ProjectVRM, http://cyber.law.harvard.edu/projectvrm/Main_Page

[5] Doc Searls, "VRM catch-up", 2008.9, http://blogs.law.harvard.edu/vrm/2008/09/06/vrm-catch-up/

[6] http://cyber.law.harvard.edu/projectvrm/VRM-vision#VRM__Capabilities_26_Standards

[7] The Higgins Project, http://eclipse.org/higgins/

[8] Social Graph, http://en.wikipedia.org/wiki/Social_graph

[9] Personal Data Model 2.0, http://wiki.eclipse.org/Persona_Data_Model_2.0

[10] http://wiki.eclipse.org/Persona_Data_Model_2.0

[11] http://wiki.eclipse.org/Personal_Data_Service_Overview

[12] Frank Manola, Eric Miller, "RDF Primer", W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-syntax/

[13] http://wiki.eclipse.org/Higgins_2.0

[14] Project Danube, http://projectdanube.org/

[15] Evan Prodromou, "What is the federated social web?", 2010.7, http://status.net/2010/07/13/what-is-the-federated-social-web

[16] OStatus, http://ostatus.org/

[17] Drummond Reed, Dave McAlpin, "Extensible Resource Identifier (XRI) Syntax V2.0", 2005.11, http://www.oasis-open.org/committees/download.php/15377

[18] Drummond Reed, Geoffrey Strongin, "The Data-web: An Introduction to XDI", 2004.4. http://www.oasis-open.org/committees/download.php/6434/wd-xdi-intro-white-paper-2004-04-12.pdf

[19] The Friend of a Friend Proejct, http://www.foaf-project.org

[20] hCard 1.0, http://microformats.org/wiki/hcard

[21] Portable Contacts, http://portablecontacts.net/

[22] http://projectdanube.org/developers/architecture/

[23] OAuth 2.0, http://oauth.net/2/

[24] OpenXRI Project, http://www.openxri.org/

[25] XDI4j 1.1, http://wiki.eclipse.org/XDI4j

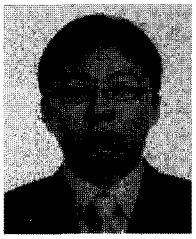
[26] The Locker Project, https://github.com/ quartzjer/Locker

[27] Telehash Project, http://telehash.org

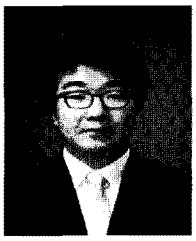
[28] singly, http://singly.com

[29] JavaScript Object Notation, http://json.org
[30] Kademlia, http://en.wikipedia.org/wiki/Kademlia
[31] WordPress, http://wordpress.com
[32] https://github.com/quartzjer/Locker
[33] The MINE! Project, http://themineproject.org/
[34] Atom, http://en.wikipedia.org/wiki/Atom
[35] Web feed, http://en.wikipedia.org/wiki/Web_feed
[36] Adriana Lukas, Mine! as VRM Infrastructure, 2008.5, https://docs.google.com/Doc?id=dgc23h2k_397cgqg3xgh&pli=1
[37] pymine, http://code.google.com/p/pymine/
[38] Django 1.0, http://www.djangoproject.com/
[39] Google App Engine, http://code.google.com/intl/ko-KR/appengine/
[40] Project Nori, http://www.projectnori.com/
[41] Personal Data Ecosystem Consortium, http://personaldataecosystem.org/
[42] Project Nori Architecture, http://www.projectnori.org/developers/architecture/

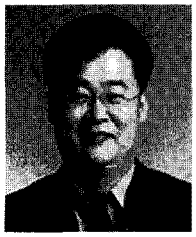
〈著者紹介〉



김 승 현(Seung-Hyun Kim)
정회원
2002년 2월 : 금오공과대학교 컴퓨터공학과 학사
2004년 2월 : 포항공과대학교 컴퓨터공학과 석사
2004년 1월~현재 : 한국전자통신연구원
관심분야 : ID 관리, 모바일 지불결제, 정보보호



김 석 현 (Seok-Hyun Kim)
정회원
2008년 2월 : 충주대학교 전자통신학과 학사
2010년 2월 : 전남대학교 정보보호협동과정 석사
2010년 7월~ 현재 : 한국전자통신연구원
관심분야 : 정보보호, 모바일 지불결제, 개인화



진 승 헌 (Jin Seung-Hun)
정회원
1993년 2월 : 숭실대학교 전자계산학과 학사
1995년 2월 : 숭실대학교 전자계산학과 석사
2004년 2월 : 충남대학교 전산학(정보보호) 박사
1996년 4월 : (주)대우통신 종합연구소 연구원
1999년 5월 : (주)삼성전자 통신연구소 전임연구원
1999년 6월~현재 : 한국전자통신연구원 인증기술연구팀장
관심분야 : 정보보호(PKI, 인증/인가기술, 프라이버시 보호기술), 모바일 지불결제, 컴퓨터/네트워크 보안