2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

PKI의 인증 경로 검증 기술

김태성, 김희선, 노종혁, 조영섭, 진승헌
한국전자통신연구원
e-mail : taesung, sezsez, jhroh, yscho, jinsh@etri.re.kr

The Mechanism of Certification Path Validation

Taesung Kim, Heesun Kim, Jonghyuk Roh, Yeongsub Cho, Seunghun Jin
Electronics and Telecommunications Research Institute

요 약

정보보호 분야의 핵심 기술인 PKI는 전자정부 및 전자상거래 전반의 응용 환경에서 정보보호의 기반구조로서 그 활용의 폭을 더욱 넓혀가고 있다. 이러한 PKI를 기반으로 하는 정보보호 서비스를 수행하기 위하여 반드시 구현되어야 하는 중요 기술 중의 하나인 인증서 검증 기술은 필수 요소임과 동시에 구현 및 수행의 복잡성으로 인해 많은 논의점들이 제기 되어 왔다. 본 논문은 현재 IETF에서 제정한 RFC2459[1] 표준 문서 및 IETF Internet-Draft[2]를 중심으로 인증서 검증 기술에 대하여 살펴보고, 최근 제시된 핫이슈들을 검토하여 인증서 검증 기술에 대한 문제점 및 이에 대한 확장 방안 및 응용에 대하여 기술한다.

1. 서론

정보보호 서비스 제공에 대한 요구의 증가에 따라 정보보호 기반 기술인 PKI 시장은 국내·외적으로 급격히 팽창하고 있다. 국내의 정보보호산업에서의 PKI 분야의 매출은 매년 2배 이상의 성장율을 보이고 있으며, 미국의 경우 PKI 시장 규모는 매년 1.5배 이상의 성장율을 나타내고 있다. 이러한 PKI 기술은 ITU-T의 X.509 인증서 표준[ ]을 근간으로 인증서 표준, 관리, 접근, 폐지와 상태조회, 타임스탬프, 정책 및 인증 실행 등의 IETF 표준 문서들을 [ ] 중심으로 활발한 연구가 진행되어 오고 있으며 이에 대한 활발한 검토가 이루어지고 있다.

X.509 인증서 기반의 PKI 시스템은 일반적으로 인증기관, 등록기관, 카드 발급 시스템, 디렉토리 서버, 키 발급 서버, 메일 서버, 시점 확인 시스템, 웹 서버로 구성되며, 인증서 관리, 등록 관리, 키 생성, 카드 발급 등의 키를 생성해서 인증서를 발급하고 폐기하기까지의 모든 일련의 과정을 수행한다. 또한, 이와 같이 시스템에서 발급되고 관리되는 인증서는 응용 레벨에서 인증서 경로 검증을 통해 확인되고 사용되므로, 인증서 검증은 인증서의 실제 활용 측면에 있어서 매우 중요한 문제가 된다. 특히, IETF에서도 인증서 검증 기술만을 수행하는 독립적인 서버에 대한 연구가 최근 활발히 진행되고 있어 이에 대한 관심은 더욱 증가하고 있다.

인증서의 인증 경로 검증에 대해서는 표준문서 RFC2459[1]에서 그 수행 절차에 대하여 언급하고 있는데, 이는 개념적인 기술로만 언급되어 있어 시스템을 실제로 구축해야 하는 구현자에게는 어려움이 많았던 것이 사실이었다. 이에 대해 최근 IETF Internet-Draft[2]에서 인증 경로에 대한 좀더 구체적인 절차를 언급하고 있어 구현자들에게는 좀더 실제적인 지침을 제시하고 있다. 그러나 여전히 이에 대한 많은 논의가 계속되고 있다.

본 논문에서는 최근 제시된 인증서 검증 경로에 대한 RFC 2459[1]와 IETF Internet-Draft[2]를 중심으로 인증 경로 검증 절차에 대해서 분석하고, 최근의 핫이슈를 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 RFC2459[1] 및 IETF Internet-Draft[2]를 바탕으로 인증 경로 검증 알고리즘에 대하여 설명하고, 3장에서는 인증서 검증 이슈에 대하여 기술한다. 그리고, 4장에서는 결론을 맺는다.

2. 인증 경로 검증 알고리즘

본 절에서는 IETF의 RFC2459 draft를 바탕으로 인증 경로 검증 알고리즘을 설명한다.

2.1 일반 사항

인증 경로 검증을 위해 필요한 인증 경로의 구성은 이 알고리즘에서 정의 하지 않고 다른 방법으로 구성 된다고 가정한다. 인증 경로 검증을 위해 구성된 인증 경로는 다음 사항을 만족해야 한다.

- { 1, ... , n-1}에 속하는 모든 인증서 x에 대해 인증서 x의 subject는 인증서 x+1의 issuer이다.
- 인증서 1은 trust anchor의해 발급 되었다.
- 인증서 n은 검증 대상 인증서이다.
- {1, ..., n}에 속하는 모든 인증서 x에 대해 인증서 x는 검증 시점에 유효해야 한다.

보통의 경우 인증서의 주체(subject)와 발급자(issuer)은 다르다. 그러나, 키 갱신이나 정책의 변경을 위해 CA가 자신에게 인증서를 발급하는 경우가 있다. 이러한 self-issued 인증서는 경로 길이를 계산하거나 이름 제한(name constraints)에 고려되지 않는다. 알고리즘에서 초기화(2.3)와 정리(2.6)는 한번 수행되고 기본 인증서 처리(2.4)는 모든 인증서에 대해 수행되며 다음 단계의 준비(2.5)는 마지막 인증서를 제외한 인증서에 대해 수행된다.

2.2 입력 파라미터

알고리즘 입력 파라미터는 certificate-path, current-date-time, trust-anchor-information, initial-policy-mapping-inhibit, user-initial-policy-set, initial-any-policy-inhibit 등이다.

certificate-path는 길이 n의 인증~경로이고 current-date-time은 현재의 날짜와 시간이며 user-initial-policy-set은 인증서 사용자가 받아 들일수 있는 정책의 집합이다. 특별히 any-policy는 어떤 정책도 관계없을 경우에 사용한다. trust-anchor-information는 발급자 이름, 공개키 알고리즘, 공개키, 공개키 파라미터로서 흔히 이것은 trust anchor의 인증서로 대치된다. initial-policy-

mapping-inhibit은 정책 매핑을 사용할 것인지를 가리킨다. initial-explict-policy는 경로가 user-initial-policy-set의 적어도 하나 이상의 정책에 대해 유효해야 하는지 가리킨다. initial-any-policy-inhibit는 인증서의 any-policy OID를 처리해야 하는지의 정보를 알려준다.

2.3 초기화

초기화 과정에서는 7 개의 입력 파라미터를 바탕으로 다음 11 개의 변수를 초기화 한다.

1) valid_policy_tree : 인증서 정책을 위한 트리이다. 단말 노드는 현재 인증서 검증 단계에서 유효한 정책을 의미한다. 트리의 깊이는 인증서 체인에서 현재 까지 처리한 인증서의 숫자와 같다. 현재 단계에서 유효한 정책이 없다면 트리는 NULL이 되고 트리가 NULL이 되면 정책 처리는 멈춘다. 트리에서 하나의 노드는 인증 경로 x에 대해 유효한 하나의 정책 OID인 valid_policy, 정책 설명인 qualifier_set, 정책 확장이 critical 인지를 나타내는 criticality_indicator, 인증 경로 x+1에서 기대되는 하나 또는 그 이상의 policy OID인 expected_policy_set로 구성된다. 이 트리의 초기 값은 [그림 1]과 같다.

오류! 연결이 잘못되었습니다.

[그림 1] valid_policy_tree 초기 값

2) permitted_subtrees : 뒤따르는 모든 인증서의 주체가 속해야 하는 이름의 집합이다. 초기 값은 모든 이름이다.
3) excluded_subtrees : 뒤따르는 모든 인증서의 주체가 속하지 말아야 하는 이름의 집합이다. 초기값은 empty이다.
4) explicit_policy : non-NULL 트리가 요구됨을 표시한다. 숫자는 이런 이 요구가 적용되기 전까지 처리해야 할 인증서(*self-issued* 인증서 제외)의 숫자이다. initial-explicit-poilicy가 true이면 0, false 이면 n+1 이다.
5) inhibit_any-policy : any-policy가 정책 매치에 사용되는 것을 의미한다. 숫자는 any-policy가 무시 되기 전까지 처리해야 하는 인증서의 숫자이다. initial-any-policy-inhibit 가 true 이면 0, false 이면 n+1 이다.
6) policy_mapping : 정책 매핑이 허용되는지를 나타낸다. 숫자는 정책 매핑이 허용 되지 않기 전까지 처리할 인증서의 숫자이다. initial-policy-mapping-inhibit가 true 이면 0, false 이면 n+1이다.
7) working_public_key_algorithm : 인증서의 서명을 검증할 공개키 알고리즘을 나타낸다. 초기 값은 trust anchor 정보에 의해 결정된다.
8) working_public_key : 인증서의 서명을 검증할 공개키이다. 초기 값은 trust anchor 정보에 의해 결정된다.
9) working_public_parameters : 인증서의 서명을 검증할 공개키의 파라미터이다. 초기 값은 trust anchor 정보에 의해 결정된다.
10) working_issuer_name : 다음 인증서의 발급자 이름이다. 초기 값은 trust anchor 정보에 의해 결정된다.
11) 11)max_path_length : 초기 값은 n 이다. 한 단계를 지날 때 마다 1 씩 감소하며, 인증서의 기본제한(basic constraint) 확장에 의해 감소한다.

오류! 연결이 잘못되었습니다.[그림 2] 기본 인증서 처리 의사 코드

2.4 기본 인증서 처리

[그림 2]는 인증서 기본 처리를 나타내는 의사코드이다. 내용을 살펴보면 VerifyBasicCertificateInformatin에서는 인증서의 기본적인 내용을 검증한다. 즉 인증서가, working_public_key, working_public_key_parameters, working_public_key_algorithm으로 서명 되었는가 확인하고, 인증서의 유효기간이 현재 시각을 포함하는가 확인하고, 현재 인증서가 폐기되거나 정지되었는가 확인하며, 인증서의 발급자 이름이 working_issuer_name인지 확인한다. CheckNameConstraint는 인증서의 주체와 subjectAltName이 permitted_subtrees에 속하는지 확인하고, excluded_subtrees에 속하지 않는가를 확인한다. valid_policy_tree에서 깊이 i-1 의 노드안에 expected_policy_set이 P-OID를 포함하고 있다면 자식 노드를 만들고 자식 노드는 valid_policy는 P-OID로, qualifier_set은 P-Q, expected_policy_set을 {P-OID}로 한다. 예를 들어 현재 인증서의 정책이 {Gold, Silver}라면 [그림 3]과 같이 자식 노드를 만든다.

오류! 연결이 잘못되었습니다.

[그림 3] valid_policy_tree 확장

i-1 노드에 매치되는 정책이 없고 깊이 i-1의 노드가 any-policy를 포함한다면 자식 노드를 만들고 valid_policy는 P-OID로, qualifier_set은 P-Q, expected_policy_set을 {P-OID}로 한다. 예를 들어 현재 인증서의 정책이 {Gold, Silver}라면 [그림 4]와 같이 자식 노드를 만든다.

오류! 연결이 잘못되었습니다.

[그림 4] 부모 노드가 any-policy일 경우 트리 확장

인증서의 정책이 any-policy를 포함하고 inhibit_any_policy가 0 보다 크다면 자식 노드가 없는 깊이 i - 1 노드에 자식 노드를 만든다. 자식 노드는 부모 노드의 expected_policy_set의 정책을 valid_policy와 expeted_policy_set으로 한다. 예를 들어 현재 인증서의 정책이 any-policy 만 있고 i-1 노드의 expected_policy_set이 {Gold, Siver}라면 [그림 5]과 같이 자식 노드를 만든다.

트리의 i-1 또는 이하의 깊이의 노드가 자식 노드가 없으면 그 노드를 삭제하고 깊이가 i-1이하이면서 자식 노드가 없는 노드가 없을 때까지 반복한다. 인증서의 정책 확장이 critical이면 criticality_indicator를 TRUE로, 그렇지 않으면 FALSE로 한다. 만약 인증서에 정책 확장이 없으면 valid_policy_tree 를 NULL로 한다. explicit_policy 0 보다 크거나 또는 valid_policy_tree가 non-NULL인지 한다. 즉, explicit_policy가 0 이고 valid_policy_tree가 NULL 이면 인증서 검증은 실패한다.

오류! 연결이 잘못되었습니다.

[그림 5] 인증서에 any-policy일 경우 트리 확장

2.5 다음 단계의 준비

인증서가 마지막 인증서가 아니면 인증 경로상의 다음 인증서 처리를 위한 준비를 다음과 같이 한다.

1) 정책 매핑 확장이 존재할 때, issuerDomainPolicy 또

는 subjectDomainPolicy에 any-policy가 존재하지 않는지 확인한다.

2) 정책 매핑이 존재하면 각 issuerDomainPolicy ID-P에 대해 다음을 수행한다. policy_mapping 이 0 보다 크면 깊이 i 인 노드의 valid_policy 가 ID-P일때 그 노드의 expected_policy_set을 ID-P에 대응 되는 subjectDomainPolicy로 변경한다. 만약 깊이 i의 어떤 노드도 valid_policy가 ID-P가 아니고 valid_policy가 any-policy이면, 깊이 i-1의 노드 중에 valid_policy가 any-policy인 노드에 자식 노드를 만들고 자식노드의 valid_policy를 ID-P로 하고, qualifier_set을 인증서 i의 any-policy의 qualifier set으로 하며, criticality_indicator를 인증서 i의 정책 확장의 critical로 하고, expected_policy_set을 ID-P에 대응 되는 subjectDomainPolicy로 한다. policy_mapping의 값이 0 이면 깊이 i 노드 중에 valid_policy가 ID-P인 노드를 지운다. 자식 노드가 없는 깊이가 i-1 인 노드를 삭제한다.
3) 인증서의 subject name을 working_issuer_name으로 한다.
4) subjectPublicKey를 working_public_key로 하라.
5) subjectPublicKeyInfo로 working_public_key_parameters 를 세팅한다.
6) subjectPublicKeyAlgorithm을 working_public_key_algorithm으로 한다.
7) 인증서에 permittedSubtrees가 있으면 현재 permitted_subtrees와 교집합(intersection)하고 인증서에 excludedSubtress가 있으면 현재 excluded_subtrees와 합집합(union)을 한다.
8) explicit_policy, policy_mapping, inhibit_any_policy의 값이 0이 아닐 경우 각 1씩 감소시킨다.
9) 인증서에 requireExplicitPolicy가 있고 explicit_policy보다 작으면 explicit_policy를 requireExplicit Policy로 한다.
10) 인증서에 inhibitPolicyMapping가 있고 policy_mapping보다 작으면 policy_mapping를 inhibitPolicy Mapping로 한다.
11) 인증서에 inhibitAnyPolicy가 있고 inhibit_any-poliy보다 작으면 inhibit_any-policy를 inhibitAnyPolicy로 한다.
12) basicConstraints으로 CA 인증서인지 확인한다.
13) self-issued 인증서가 아니고 max_path_length이 0 보다 크면 1을 감소하라.
14) 인증서에 pathLengthConstraint가 있고 max_path_length보다 작으면, max_path_length를 pathLength Constraint로 한다.
15) key usage가 있으면 keyCertSign bit가 세트되어 있는지 확인한다.
16) 다른 critical 확장을 인식하고 처리하고, 인식된 다른 non-critical 확장을 처리한다.

2.6 정리

End entity의 인증서 처리를 마무리 하기 위해서는 인증서 n에 대하여 다음을 수행한다. 인증서 n이 self-issued가 아니고 explicit_policy가 0이 아니면 explicit_ policy를 1 감소시키다. 또한, 정책 확장이 있고 requireExplicitPolicy가 0 이면 , explicit_policy를 0 으로 한다. 마지막으로 valid_policy_tree와 user-initial-policy-set의 교집합(intersection)을 [그림 6]과 같이 구한다. *[그림 6]에서 valid_policy_tree가 NULL 이면 교집합은 NULL이고 valid_policy_tree가 NULL이 아니고, user-initial-policy-set이 any-policy 이면 교집합은 valid_policy_tree 이다. valid_policy_tree가 NULL이 아니고, user-initial-policy-set이 any-policy가 아니면 부모의 노드 중에 valid_policy가 any-policy인 노드들의 집합을 valid_policy_node_set이라 한다. valid_policy_node_set의 노드의 valid_policy가 any-policy가 아니고 user-initial-policy-set에 속하지 않으면 이 노드와 모든 자식 노드를 삭제한다. 깊이가 i-1이하이면서 자식 노드가 없는 노드를 삭제하고 깊이가 i-1 이하이면서 자식 노드가 없는 노드가 없을 때까지 반복한다. 마지막에 남은 valid_policy_tree가 교집합이다. explicit_policy가 0 보다 크거나, 또는 valid_policy_tree가 NULL이 아니면 검증은 성공한다.

3. 인증서 검증 이슈

RFC 2459 draft의 인증 경로 검증에 있어 이슈는 크게 두 가지이다. 첫째 인증서 경로에서 처음 인증서가 trust anchor의 self-signed 인증서이어서는 안된다는 것이다. 여기서 self-signed 인증서의 공개키에 해당되는 개인키에 의해 서명된 인증서이고, self-issued 인증서는 발급자와 주체가 동일한 인증서를 지칭하므로 구분할 필요가 있다. trust anchor의 정보는 초기화 과정에 이미 들어가 있으므로 trust anchor의 self-issued 인증서는 가능하지만 self-signed 인증서는 인증경로에 있을 필요가 없다. 둘째, 인증 경로에 어떤 CA의 self-signed 인증서도 들어올 수 없어야 한다는 것이다. 현재 2459의 draft는 self-signed 인증서가 인증 경로에 포함되어야 한다거나 포함되지 말아야 한다고 명시하지 않고 다분히 모호하게 포함 할 수도 있다고 표현되어 있다. 예를 들어, 어떤 CA가 인증 경로 검증에 self-signed 인증서가 포함 될것을 예상하고 self-signed 인증서에 이름제한(name constraint)를 넣었다면, self-signed 인증서에 대한 인증 경로의 포함 여부에 따라 인증 경로 검증 결과가 다르게 되는 문제가 있다. 또한, 인증서는 디렉토리를 통해 가져오게 되므로 self-signed 인증서를 인증 경로에 포함시키는 것은 구현에 부담이 될 수 있다. 그리고, X.509[3]는 인증 경로에 self-signed 인증서를 포함하지 못하도록 하고 있으므로 상호운용성이 떨어지는 결과를 가져온다.

오류! 연결이 잘못되었습니다.

[그림 6] valid_policy_tree와 user-initial-policy-set의 교집합(intersecion)을 구하는 의사코드

4. 결론

본 논문에서는 인증서의 인증 경로 검증 알고리즘과 인증서 검증 이슈에 대하여 살펴보았다. 인증 경로 검증 알고리즘은 RFC2459[1]와 IETF의 Internet-Draft[2]를 기준으로 경로 검증 처리의 입력 파라미터, 초기화, 기본 인증서 처리 및 다음 차례의 인증서 처리 준비 및 정리 단계를 알아보았다. 인증서 경로 검증에 관한 이슈로는 인증서 경로에서 처음 인증서는 trust anchor의 self-signed 인증서가 아니어야 한다는 것과 인증 경로에 어떤 CA의 self-signed 인증서도 들어올 수 없어야 하는 것이 언급되었다.

인증서에 대한 인증 경로 검증 기법은 인증서가 실제적으로 활용되는 응용에서 매우 중요한 부분을 차지한다. 새롭게 제시된 보다 구체적인 인증 경로 검증 알고리즘은 기존에 제시된 표준 문서에 대한 다양한 해석의 폭을 좁혀 줌

과 동시에 좀더 구체적인 구현 방안을 제시해 주었다. 반면, 아직 Internet-Draft 단계이므로 앞서 제시되었던 이슈를 중심으로 계속적으로 거론되는 문제점들은 수정되어 표준에 반영되어야 하겠다. 이에 따라, 인증서 검증 모듈에도 표준에 반영되는 추가 사항들이 적극 반영되어 구현되어야 한다.

참고문헌

[1] Housley, R., W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 2459, January 1999.

[2] Housley, R., W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile draft-ietf-pkix-new-part1-08", July 2001

[3] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997