XML-Sicherheit in der Datenbank

Dominik Schadow, Trivadis GmbH

Mit der XML-Sicherheit gibt es speziell auf die Bedürfnisse und Möglichkeiten der Extensible Markup Language ausgerichtete Varianten der digitalen Signaturen und Verschlüsselung. Dieser Artikel bietet eine Einführung in die XML-Sicherheit und zeigt gleichzeitig, welch geringe Auswirkungen der damit erzielte Sicherheitsgewinn auf eine Speicherung in der Datenbank hat.

Die Extensible Markup Language (XML) ist heute nahezu überall anzutreffen. Ob als Konfigurationsdatei, als anwendungs- und plattformunabhängiges Speicherformat oder in der Datenbank – XML erscheint allgegenwärtig. Es gibt kaum eine Applikation, vom einfachen Editor bis zur komplexen Datenbank, die nicht zumindest lesend auf XML zugreift. Web Services tragen ebenfalls ihren Teil zur Verbreitung bei.

Die Oracle Datenbank ermöglicht dabei neben dem Speichern als normalen Text auch das Speichern von XML in dessen hierarchischem Format als XMLTYPE. Einzige Bedingung: Das XML muss wohlgeformt sein. Mit dieser Speicherungsform können anschließend selbst Indizes auf einzelnen Elementen oder Attributen erstellt werden. Ebenso lassen sich mit XQuery, der XML-Abfragesprache, gezielt Abfragen auf einzelne Elemente oder Attribute durchführen. Neben der relationalen Speicherung etabliert sich hier eine zweite, streng hierarchische Speicherungsform für XML in der Datenbank.

Unabhängig vom Speicherort ist XML prinzipiell immer leicht von Menschen und Maschinen lesbar und lässt sich meist auch einfach erstellen. Und genau hier kann bei sensiblen Informationen, wie sie heute bereits in großen Mengen in einer relationalen Datenbank (verschlüsselt) gespeichert werden, ein Problem entstehen.

XML-Sicherheit

Natürlich lässt sich XML, genau wie jede andere digitale Information auch, beim Transport beispielsweise einfach mit https sichern. Spätestens in der Datenbank können dann Daten, vom BLOB bis hin zu XML, etwa mit Oracle Transparent Data Encryption verschlüsselt werden.

Der Datensicherheit ist damit Genüge getan, allerdings geht durch die Verschlüsselung die XML-Struktur verloren. Ein außerhalb der Datenbank auf herkömmliche Weise verschlüsseltes XML kann überhaupt nicht mehr als XMLTYPE gespeichert werden, das XML ist schlicht nicht mehr wohlgeformt. Sofern man hierbei überhaupt noch von XML sprechen kann: Beim Verschlüsseln ging gleichzeitig auch die XML-Struktur völlig verloren. Davon abgesehen steht mit https lediglich eine sogenannte Punkt-zu-Punkt (oder auch transportgebundene) Sicherheit zur Verfügung. Digital signiert sind die Daten so auch nicht. Vor allem muss die Sicherheit der Daten an zwei unterschiedlichen Punkten und mit zwei unterschiedlichen Technologien sichergestellt werden: beim Transport und bei der Speicherung in der Datenbank.

Die auf offiziellen Empfehlungen des World-Wide-Web-Consortiums basierende XML-Sicherheit behebt diese und weitere Nachteile der herkömmlichen Sicherungsformen für XML-basierte Formate: Die XML-Struktur bleibt erhalten und steht gleichzeitig von Anfang bis Ende zur Verfügung (Ende-zu-Ende-Sicherheit oder auch informationsgebundene Sicherheit).

Im Gegensatz zu https wird also die Nachricht (Information) und nicht der Transport gesichert. Die Transportsicherung kann zusätzlich immer noch durchgeführt werden, ist aber an sich unnötig. Außerdem bleibt, im Gegensatz zu allen anderen Sicherungsformen, auch die XML-Struktur erhalten, wenn auch mit einigen Einschränkungen (siehe unten).

Feingranulare Sicherung

Bisherige Sicherungsformen wie die Signierung / Verschlüsselung mit symmetrischen, asymmetrischen oder hybriden Krypto-Algorithmen, gehen mit dem streng hierarchischen XML-Format nicht richtig um. Sie arbeiten in der Regel nach dem Motto „Ganz oder gar nicht“ – es wird entweder alles signiert / verschlüsselt oder eben gar nichts. Das vollständige Signieren und / oder Verschlüsseln bedeutet oftmals jedoch einen gewaltigen Overhead, sollen beispielsweise bei einer Bestellung doch nur die sensiblen Kreditkarten-Informationen verschlüsselt werden. Trotz aller Fortschritte bei Hard- und Software sind die Verschlüsselung sowie die digitale Signatur immer noch ein relativ langsamer Vorgang. Die Entschlüsselung und Verifizierung auf der Gegenseite entsprechend auch.

Mit der XML-Sicherheit lässt sich dagegen sehr feingranular festlegen, welche Dokument-Fragmente gesichert werden sollen und welche nicht. Bis hinunter zum Element-Inhalt lässt sich so bestimmen, welche Dokumentteile in eine Signatur oder Verschlüsselung einbezogen werden. Gleichzeitig ermöglicht diese feingranulare Sicherung das Verwenden unterschiedlicher Schlüssel für verschiedene Dokument-Fragmente: Der Händler besitzt beispielsweise nur noch den Schlüssel für den Bestellungs-Dokumentbereich und die Bank den für den Zahlungs-Dokumentbereich. Folglich können diese

Teilnehmer nur die für sie relevanten Informationen entschlüsseln.

Übrigens lassen sich mit der XML-Verschlüsselung und den XML-Signaturen auch binäre Daten sichern. Diese Daten werden base-64-kodiert in das XML-Dokument eingebettet. Die Folge ist allerdings eine um rund 33 Prozent größere Datenmenge.

Bei all diesen vielen Möglichkeiten führen die W3C-Empfehlungen keinerlei neue Sicherheitsmechanismen ein. Sie legen vielmehr fest, wie XML-Dokumente oder XML-Dokument-Fragmente signiert beziehungsweise verschlüsselt und wie diese Sicherheitselemente in das XML-Dokument eingebettet werden. Die Empfehlungen basieren daher auf bekannten, und vor allem bewährten, kryptografischen Mechanismen und Prinzipien. Das W3C gibt hier eine Reihe von kryptografischen Algorithmen vor, die von allen standardkonformen Implementierungen integriert werden müssen oder können. Weitere können von der jeweiligen Implementierung frei hinzugefügt werden, worunter dann allerdings die Interoperabilität mit anderen Herstellern leiden kann.

XML-digitale-Signaturen

Wie bereits angeführt, unterscheiden sich XML-digitale-Signaturen von herkömmlichen Signaturen grundsätzlich erst einmal dadurch, dass sich diese Signaturen der XML-Struktur „bewusst“ sind und diese auch mit Signatur erhalten bleibt. Ein Parsen oder eine anderweitige automatische Verarbeitung sind somit trotz eingefügter Signatur möglich.

Um möglichst vielen Anforderungen gerecht zu werden, ist die gesamte Empfehlung zu den digitalen Signaturen relativ komplex geraten. Grundsätzlich existieren digitale Signaturen für XML erst einmal in drei Richtungen (die auch kombiniert werden können): enveloped, enveloping und detached. Im Grunde geht es dabei nur um die unterschiedlichen Positionen von Signatur und signiertem Inhalt im XML.

Bei einer *enveloped signature* stellt das zu signierende Dokument das Eltern-

Element zum Signature-Element dar. Das Signature-Element ist somit vom zu signierenden Dokument umgeben.

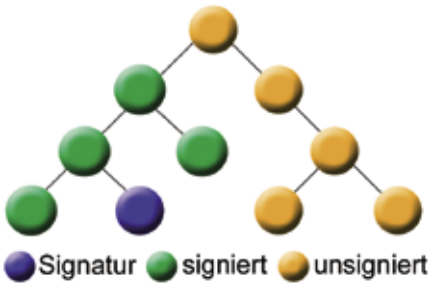

Abbildung 1: enveloped signature

Werden die zu signierenden Daten dagegen als Kind-Elemente der Signatur dargestellt, wird dies als *enveloping signature* bezeichnet. Der signierte Inhalt steht bei dieser Signaturvariante als Kind-Element innerhalb des Signature-Elements.

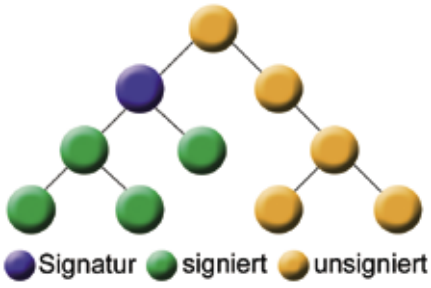

Abbildung 2: enveloping signature

Bei einer *detached signature* schließlich gibt es keine Eltern-Kind-Beziehung zwischen Signatur und Dokument. Das Original-Dokument ist meist eine separate Ressource und wird auch nicht verändert. Der signierte Inhalt kann sich aber grundsätzlich im selben XML-Dokument wie die Signatur (aber ohne Eltern-Kind-Beziehung) oder aber in einem anderen Dokument befinden.

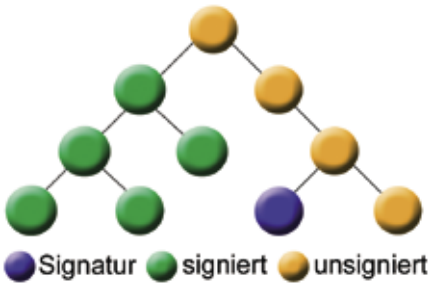

Abbildung 3: detached signature

Die XML-Signatur hat, egal welcher Typ angewendet wird, immer den folgenden Aufbau (? erlaubt das null- bis einmalige, + das mindestens ein- bis mehrmalige und * das null- bis mehrmalige Vorkommen als Attribut oder Element):

```
<ds:Signature ID?>
  <ds:SignedInfo>

<ds:CanonicalizationMethod/>
   <ds:SignatureMethod/>
   (<ds:Reference URI?>
    (<ds:Transforms/>)?
    <ds:DigestMethod/>
    <ds:DigestValue/>
   </ds:Reference>)+
  </ds:SignedInfo>
  <ds:SignatureValue/>
  (<ds:KeyInfo/>)?
  (<ds:Object ID?/>)*
</ds:Signature>
```

Aus Platzgründen kann hier leider kein vollständiges Beispiel eines signierten XML-Dokuments gezeigt werden. Hier sei, ebenso wie für eine vollständige Beschreibung aller Elemente und Attribute, auf die offizielle W3C-Empfehlung oder auf das Basics Tutorial unter www.xml-sicherheit.de verwiesen.

Zu beachten ist, dass sämtliche verwendeten Algorithmen über ein Algorithmen-Attribut im XML-Dokument selbst bekanntgegeben werden. Auch die durchgeführten Transformationen und die Kanonisierung werden angegeben. Ebenso können Informationen zum verwendeten Schlüssel über das Key Info Element integriert werden. Die verifizierende Anwendung lässt sich damit leicht dieselben Transformationen und Berechnungen durchführen und die Signatur bestätigen. Fehlen diese teilweise optionalen Informationen, muss der verifizierenden Anwendung bekannt sein, welche Algorithmen verwendet wurden. Der Vorteil eines kleineren XML-Dokuments führt so zu einer sehr viel größeren Abhängigkeit zwischen der signierenden und der verifizierenden Applikation.

Vom ID-Attribut der Signatur sollte, auch wenn es optional ist, immer Gebrauch gemacht werden. Die Verifikation wird bei mehreren Signaturen bedeutend einfacher wenn Signaturen eindeutig identifiziert werden können.

Einem grundsätzlichen Problem kommt bei den digitalen Signaturen mit XML besonders große Bedeutung zu: Trotz logischer Gleichheit der reinen Dokumentdaten lassen sich Veränderungen am physischen Dokument durch die Verarbeitung, beispielsweise durch verschiedene Parser oder beim Datenbank-Zugriff über unterschiedliche Treiber, nicht vollständig verhindern. Um dadurch ungültige Signaturen auszuschließen, wird vor dem Signieren ein Kanonisierungs-Algorithmus auf das XML-Dokument angewendet. Hiermit werden syntaktische Unterschiede – wie beispielsweise der Zeichensatz, Zeilenumbrüche, leere Tags und die Reihenfolge der Attribute – von ansonsten semantisch identischen XML-Dokumenten normalisiert. Vor der Verifizierung der Signatur wird wieder der gleiche Kanonisierungs-Algorithmus (es gibt mehrere Versionen und Varianten) angewendet. Wie alle verwendeten Algorithmen (auch die zum Signieren) wird dazu auch der Kanonisierungs-Algorithmus im XML-Dokument gespeichert.

XML-Verschlüsselung

Auch bei der XML-Verschlüsselung bleibt die XML-Struktur erhalten, womit natürlich nicht das ursprüngliche XML-Dokument im Klartext gemeint ist. Es besteht aber die Möglichkeit, das ursprüngliche Wurzelelement im Klartext beizubehalten. Alternativ wird das Wurzelelement der XML-Verschlüsselung übernommen. In jedem Fall ist auch das verschlüsselte XML wohlgeformt und kann von einem Parser geparst und weiterverarbeitet werden. Wie bei den XML-digitalen-Signaturen auch, lassen sich bei der Verschlüsselung mit XML gezielt einzelne Elemente oder Element-Inhalt verschlüsseln. Grundsätzlich besitzt ein verschlüsseltes XML-Fragment oder XML-Dokument immer den folgenden Aufbau (? erlaubt wiederum das null- bis einmalige, + das mindestens ein- bis mehrmalige und * das null- bis mehr-

malige Vorkommen als Attribut oder Element):

```
<xenc:EncryptedData Id?
Type? MimeType? Encoding?>
  <xenc:EncryptionMethod/>?
  <ds:KeyInfo>
    <xenc:EncryptedKey/>?
    <xenc:AgreementMethod/>?
    <ds:KeyName/>?
    <ds:RetrievalMethod/>?
    <ds:*/>?
  </ds:KeyInfo>?
  <xenc:CipherData>
    <xenc:CipherValue/>?
    <xenc:CipherReference
URI?/>?
  </xenc:CipherData>

>xenc:EncryptionProperties/>?
</xenc:EncryptedData>
```

Aus Platzgründen kann auch hier leider kein vollständiges Beispiel eines verschlüsselten XML-Dokuments gezeigt werden.

Gleichermaßen wie bei den digitalen Signaturen ist auch hier zu bemerken, dass verwendete Algorithmen im XML-Dokument eingefügt werden und optional Informationen zum verwendeten Schlüssel integriert werden können. Key Info greift, wie der ds-Namespace schon andeutet, dabei auf das Key-Info-Element der XML-digitalen-Signaturen zurück.

Auch bei der Verschlüsselung sollte das optionale ID-Attribut stets angegeben werden. Ebenso machen die drei anderen optionalen Attribute Type, MimeType und Encoding die weitere Verarbeitung (meist wohl die Entschlüsselung) einfacher und sollten daher immer im XML enthalten sein.

Welche Art von Verschlüsselungsalgorithmus verwendet wird, also symmetrisch, asymmetrisch oder hybrid, bleibt dabei völlig dem Anwender überlassen. Wie er den möglicherweise symmetrischen Schlüssel auf sichere Art mit seinem Gegenüber austauscht auch.

XML in der Datenbank

Derartig gesicherte XML lässt sich ohne Weiteres in die Oracle Datenbank einfügen. Da das XML in jedem Fall wohlgeformt ist, kann und soll der XMLTYPE als Datentyp verwendet werden. Bei einem binären, auf der Abstract Syntax Notation One (ASN.1)-basierten, Signatur- oder Verschlüsselungsformat ist das nicht möglich. Wer bisher in der Datenbank nicht den XMLTYPTE-Datentyp verwendet hat, kann jetzt trotz gesichertem XML auf diesen Typ migrieren und von dessen erweiterten XML-Manipulationsmöglichkeiten profitieren.

Sofern keine Validierung beim Einfügen in die Datenbank durchgeführt wird, sollten durch das gesicherte XML normalerweise keine Änderungen beim

Datenzugriff notwendig werden. Selects/Inserts und Updates lassen sich wie bisher weiterverwenden, lediglich die verarbeitende Applikation muss natürlich entsprechend angepasst werden.

Ein ungesichertes XML-Dokument, das bisher zum Beispiel über

```
INSERT INTO invoice VALUES
(,<?xml version="1.0"
encoding="UTF-8"?>
<invoice date="2008-08-
21">...</invoice>');
```

in die Datenbank eingefügt wurde, kann damit auch als verschlüsseltes XML

```
INSERT INTO invoice VALUES
(,<?xml version="1.0"
encoding=»UTF-8»?>
<xenc:EncryptedData
xmlns:xenc=»http://www.
w3.org/2001/04/xmlenc#»
 Id=»invoice» Type=»http://
www.w3.org/2001/04/
xmlenc#Element»>
...</xenc:EncryptedData>');
```

ohne Probleme verwendet werden.

Lediglich beim Insert oder Update in einer XMLTYPE-XMLSCHEMA-Spalte gilt es zu beachten, dass das verschlüsselte XML-Dokument höchstwahrscheinlich nicht einmal mehr das Wurzelelement mit dem ursprünglichen XML-Dokument gemeinsam hat. Bei signierten XML-Dokumenten müssen entsprechend die eingefügten Signatur-Elemente im XML-Schema berücksichtigt werden. Allerdings erscheint eine Validierung von signiertem oder verschlüsseltem XML in der Datenbank ohnehin nicht mehr besonders sinnvoll. Hier ist es empfehlenswert, das XML unmittelbar vor dem Sichern, somit außerhalb der Datenbank zu validieren und die Daten ohne weitere Überprüfung direkt in die Datenbank zu übertragen.

Gleichzeitig muss beachtet werden, dass die Datenmenge, je nach den ursprünglichen XML-Daten, durchaus beträchtlich größer wird. Vor allem bei der Integration von binären Daten in base-64-Kodierung wird sich der erhöhte Speicherbedarf bemerkbar machen. XML gilt grundsätzlich als „geschwätziges" Format, mit der Sicherung nimmt das verständlicherweise noch etwas zu.

Herausforderungen

Die XML-Sicherheit stellt die Anwender und vor allem auch die Entwickler vor neue Herausforderungen. Auf der einen Seite erfordern die umfangreichen Empfehlungen eine sorgfältige und komplexe Implementierung. Mit den Empfehlungen zu den digitalen Signaturen und zur Verschlüsselung alleine ist es auch nicht getan. Auf der anderen Seite spielen die W3C-Empfehlungen zur Kanonisierung und zum Key Management (XKMS, die XML-Key-Management-Specification) sowie grundlegende Empfehlungen wie die XPath Expressions und andere eine Rolle. Besonders der Kanonisierung kommt, wie oben bereits kurz angesprochen, eine große Bedeutung zu.

Zum anderen entstehen durch XML und die XML-Sicherheit aber auch neue Angriffsformen und Probleme. Stellvertretend für viele sei hier das Präsentationsproblem angeführt. Grundsätzlich existiert das auch bei herkömmlichen Signaturen, ist also nicht erst durch XML entstanden. Dabei wird dem Anwender beispielsweise bei einer Bestellung ein Preis von 100 Euro zum Signieren angezeigt, in Wirklichkeit lautet der Preis aber 1.000 Euro. In gutem Glauben signiert der Anwender das XML-Dokument und gibt die Bestellung auf. Bei XML wird das besonders einfach durch die Verwendung der Extensible Stylesheet Language Transformations (XSLT). Diese in der Regel separaten Datei(en) werden oftmals nicht mit signiert, können von einem Angreifer also ohne Auswirkung auf die Signatur manipuliert werden. Bei der Implementierung steht man hier vor der Wahl, alle an der Signatur beteiligten Dateien einzubeziehen (das können durch Imports eine ganze Menge sein) oder dem Anwender zur Sicherheit das unformatierte XML anzuzeigen.

Fazit

Für das offene Datenformat XML bestehen, wie gezeigt, mehrere Sicherungsmöglichkeiten: ob direkt in der Datenbank mit dem Oracle-eigenen Transparent Data Encryption, transportgebunden, informationsgebunden mit herkömmlichen Mitteln oder den W3C-Empfehlungen.

Bei der Entscheidung für die W3C-Empfehlungen sorgen sicherlich die umfangreichen Empfehlungen zu den digitalen Signaturen und der Verschlüsselung mit XML für einen anfangs hohen Aufwand. Komplexität und Sicherheit haben noch nie gut zusammengepasst, bei der Implementierung sollte man daher besondere Vorsicht walten lassen. Anschließend verfügt man aber über die informationsgebundene Sicherheit von der Entstehung der Daten bis zur Speicherung in der Datenbank. Auch an die Sicherung des Transportweges muss man dann kaum mehr einen Gedanken verschwenden. Gleichzeitig ist man anwendungs- und plattformneutral aufgestellt.

Abhilfe in Sachen Komplexität verspricht die, zumindest nach derzeitigem Stand der XML Signature Working Group, kommende Version 2 der Signatur-Empfehlung: Diese soll einen modulareren Aufbau besitzen und den bewussten Verzicht auf einzelne Teile ermöglichen. Vor 2010 ist damit allerdings nicht zu rechnen.

Weitere Informationen

- XML Signature: http://www.w3.org/TR/xmldsig-core/
- XML Encryption: http://www.w3.org/TR/xmlenc-core/
- XML-Sicherheit: http://www.xml-sicherheit.de

Kontakt:
Dominik Schadow
dominik.schadow@trivadis.com