Kazienko P.: Bezpieczeństwo dokumentów XML. VII Krajowa Konferencja Zastosowań Kryptografii ENIGMA 2003, Warszawa 12-15 maja 2003, s. K-191-203.

Bezpieczeństwo dokumentów XML

Przemysław Kazienko
Zakład Systemów Informacyjnych, Wydział Informatyki i Zarządzania, Politechnika Wrocławska
Wyb. Wyspiańskiego 27, 50-370 Wrocław
kazienko@pwr.wroc.pl, http://www.pwr.wroc.pl/~kazienko

Streszczenie

W pracy omówiono dwa najważniejsze standardy wykorzystywane w celu zapewnienia ochrony dokumentom XML: XML Signature oraz XML Encryption. Umożliwiają one kryptograficzne zabezpieczanie zarówno całych dokumentów XML jak i ich fragmentów (treści elementów lub całych elementów ze znacznikami i atrybutami). Dane źródłowe oraz ich postać zaszyfrowana mogą być umieszczane w jednym dokumencie lub w zupełnie osobnych zbiorach, nawet rozproszonych w sieci Internet. Dopuszcza się stosowanie zarówno wielu, standardowych algorytmów kryptograficznych jak również własnych, opracowanych przez użytkownika. Ta elastyczność ma kluczowe znaczenie w elektronicznej wymianie danych, w której język XML zaczyna dominować.

Oba standardy zostały zatwierdzone w 2002 r. i dzięki temu wspierają pozostałe standardy z rodziny XML: schematy XML Schema, przestrzenie nazw, język wskazań XPath, itd. W referacie zaprezentowano zasadnicze możliwości języków XML Signature oraz XML Encryption, wraz przykładami ich zastosowań. Wskazano również na problemy jakie mogą pojawić się przy ich wykorzystaniu, zwłaszcza w sieci Internet.

1. Język XML i jego charakterystyka

Rozszerzalny język znaczników – XML (*eXtensible Markup Language*) został zatwierdzony jako standard (Rekomendacja organizacji WWW Consortium) w lutym 1998 roku. Powstał on w odpowiedzi na wciąż wzrastające potrzeby dynamicznie rozwijającego się systemu WWW, w którym jako języka opisu dokumentów – stron, powszechnie używano języka HTML (*HyperText Markup Language*). Zarówno XML jak i HTML są językami znaczników. We wszystkich językach znaczników, tekstowa informacja jest umieszczana pomiędzy **znacznikami** (*tag, markup*): początkowym i końcowym. Znaczniki i to, co się znajduje pomiędzy nimi tworzą **element**. Typowe elementy zawierają tekstową **treść** lub inne elementy potomne – **podelementy**. Gdy elementy są puste, wtedy mogą przyjmować postać pojedynczego znacznika elementu pustego. Istnieje także — stosunkowo rzadko wykorzystywana — mieszana zawartość elementów, w której treść tekstowa jest przeplatana podelementami. Każdy dokument zawiera jeden element główny – **korzeń**. Ponieważ elementy zawierają się w innych elementach (wyłącznie w całości), więc dokument można traktować jako **hierarchię elementów**. W znaczniku początkowym lub znaczniku elementu pustego można umieszczać **atrybuty**, które posiadają tekstową **wartość**. Charakterystyka ta dotyczy zarówno języka HTML jak i XML [Kaz02f].

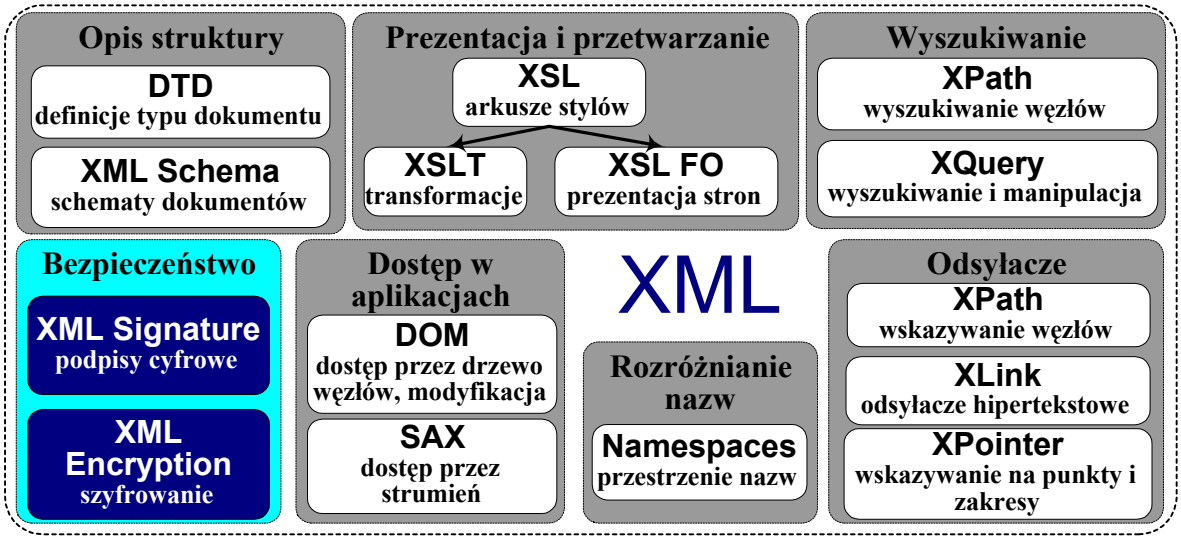
Główną cechą różniącą język XML od HTML jest to, że w XML autor może tworzyć własne elementy. Dzięki temu język XML jest **metajęzykiem**, tzn. za jego pomocą można definiować własne języki. Do takiego definiowania służą dwa mechanizmy: DTD (*Document Type Definition*) oraz schematy XML Schema. Te ostatnie, istniejące od 2 lat umożliwiają tworzenie bardzo precyzyjnych opisów struktury dokumentu [Kaz02b]. Typowe dokumenty XML zawierają merytoryczną informację na jakiś temat. Aby umożliwić jej wykorzystanie w środowisku WWW już pod koniec 1999 roku zatwierdzono standard XSLT (*XSL Transformation*), który umożliwia tworzenie (także jako dokumenty XML) opisów sposobu przekształcenia dokumentu XML danego typu (o danej strukturze) w innym dokument tekstowy, w szczególności do: HTML; XSL FO (*XSL Formatting Object*) będącego XML-owym opisem wyglądu strony wydruku; zwykłego tekstu, bez znaczników; drugiego dokumentu XML, ale o innej niż dokument źródłowy strukturze; XML-owego rysunku wektorowego w formacie SVG (*Scalable Vector Graphics*), itd.. Aby tego dokonać należy posiadać dowolny proces XSLT, w który obecnie wyposażone są m.in. wszystkie przeglądarki internetowe.

2. Rodzina języków XML

XML nie jest pojedynczym standardem, ale ich rodziną [Kaz02d], które dopiero łącznie stanowią o jego sile (rys. 1). Wśród nich znajdują się także standardy zapewniające bezpieczeństwo dokumentom XML: XML Signature i XML Encryption [Kaz02a]. Wykorzystuje się w nich w zasadzie wszystkie pozostałe standardy, tj.:

- XML Schema i DTD do opisu języków XML Signature i XML Encryption a także konkretnych typów dokumentów związanych z ich zastosowaniami,
- przestrzenie nazw (*namespaces*) do rozróżniania nazw danego języka; także nazw związanych z podpisami i szyfrowaniem, co jest wykorzystywane w XML Encryption do przechowywania informacji o kluczu pochodzących z języka XML Signature) – patrz atrybut **`xmlns`** w dalszych przykładach
- XPath do wskazywania np. fragmentów dokumentu XML, które są szyfrowane lub podpisywane

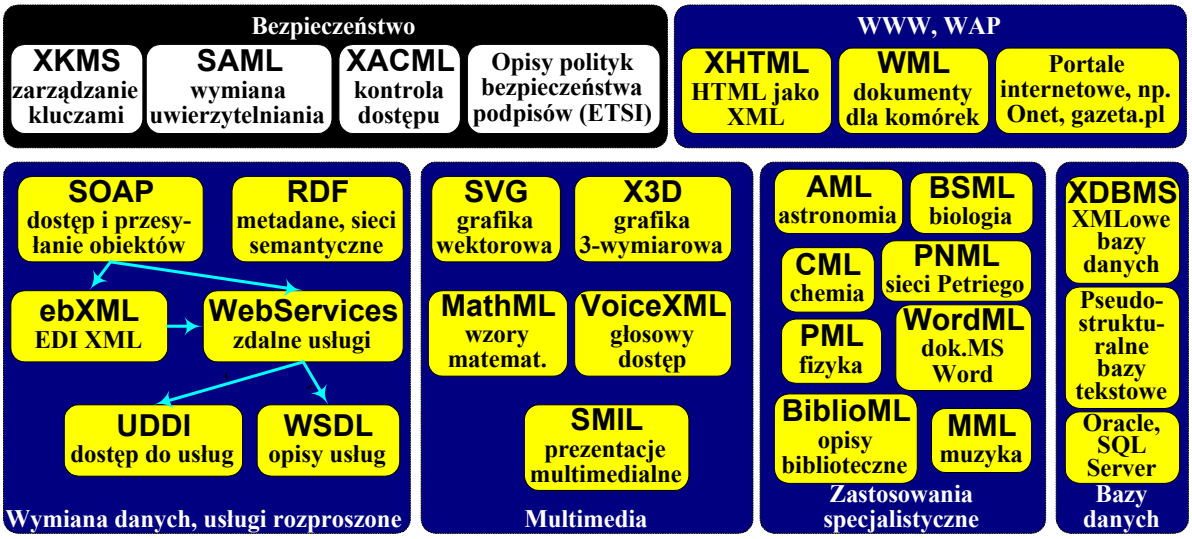
- implementacje interfejsów dostępu z poziomu aplikacji: DOM lub SAX
- XSL do prezentacji danych po zdeszyfrowaniu.



Rys. 1. Dodatkowe standardy związane z językiem XML

3. Obszary zastosowań języka XML

Język XML jest już powszechnym standardem w środowisku internetowym i zajmuje coraz ważniejsze miejsce w elektronicznej wymianie danych. Obszary jego zastosowań są obecnie bardzo szerokie a ich przegląd (oczywiście niepełny) zawarto na rys. 2. W zakresie bezpieczeństwa (poza omówionymi dalej standardami XML Signature i XML Encryption) można wymienić [Kaz02c]: język XKMS przewidziany do zarządzania kluczami kryptograficznymi, SAML — do wymiany pomiędzy systemami informacji uwierzytelniających użytkowników i obiekty, XACML – opisy list dostępu czy opisy polityk bezpieczeństwa dla certyfikatów [ETSI02b].



Rys. 2. Wybrane obszary zastosowań języka XML

4. Problem bezpieczeństwa dokumentów XML

Ze względu na duże rozpowszechnienie dokumentów XML zagadnienie zapewnienia odpowiedniego poziomu ich bezpieczeństwa stało się bardzo ważkim problemem, szczególnie tam, gdzie dokumenty te są przekazywane poprzez publiczną sieć, np. Internet. Oczywiście tak jak i dla innych dokumentów tekstowych transportowanych elektronicznie, można wykorzystać znane, standardowe mechanizmy zabezpieczania transmisji (SSL, WTLS, VPN, VLAN) lub zabezpieczania dokumentów, jak pocztowe S/MIME czy PGP. Dotyczą one jednak całych dokumentów XML, zaś dodatkowo w przypadku zabezpieczania transmisji, po jej zakończeniu, dysponujemy jedynie niezaszyfrowaną postacią danych, co uniemożliwia jakąkolwiek późniejszą weryfikację.

Nowe języki XML Signature i XML Encryption dostarczają mechanizmów specjalizowanych przeznaczonych przede wszystkim dla dokumentów XML. Umożliwiają one zabezpieczanie zarówno całych dokumentów jak i ich części (elementów lub zawartości elementów – bez znaczników). Dzięki temu ochrona może być selektywna [Kaz02c]. Bezpieczna postać informacji może być zintegrowana z dokumentem źródłowym lub z nim rozdzielona (ewentualnie połączona poprzez odpowiednie odesłania). W różny sposób można także podać informację o kluczu deszyfrującym: w postaci zaszyfrowanej, jako identyfikator certyfikatu, wskazanie na miejsce w sieci, klucz publiczny (np. certyfikat) w postaci jawnej lub zaszyfrowanej (XML Encryption).

5. XML Signature

Standard XML Signature został zatwierdzony w lutym 2002 [Sig02] i stosunkowo szybko wiele firm, organizacji i osób zaczęło go stosować. Cały podpis w języku XML Signature jest umieszczony w pojedynczym elemencie **`Signature`**. W jednym dokumencie może być wiele elementów **`Signature`**, dzięki czemu możliwe jest łączenie podpisów. Oto przykład dokumentu XML zawierającego podpis:

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo>
  <CanonicalizationMethod
    Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
  <Reference URI="http://przyklad.pl/xml/do-podpisu.xml">
   <Transforms>
    <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#base64"/>
   </Transforms>
   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                                    <!-- skrót z podpisywanych danych -->
   <DigestValue>60NvZvtdTB+7UnlLp/H24p7h4bs=</DigestValue>
  </Reference>
 </SignedInfo>
 <SignatureValue>OsH9A1jTNLmEldLmsPLlog6Gdw4YV8SiqD96GwYLAfMBqbk5o3waOg==
 </SignatureValue>       <!-- zaszyfrowany skrót z SignedInfo - podpis  -->
 <KeyInfo>
  <KeyValue>             <!-- parametry klucza publicznego -->
   <DSAKeyValue>
    <P>imup6lmki4rAmUstKb/xdBRMWNtQ+pDN97ZnLA9X3lKbkEHtYFyjQ3uActgVSJ75iVRu
       Kxz4Cb5RzVm25EaKmKq8rif1MtBIi6jjDJxmIdNaEKG9zVTf9giJx1N9I0t3oh1fAVZD
       SrzKzJGQ2WvDfIfFHdJMtB3C0VKGmLZR7Xk=</P>
    <Q>xDve3j7sEnh4rIzM5gK+5/gxxFU=</Q>
    <G>NLugAf6IZJxo3BCOi5yrGEVwtlEzXcnndXhd0Tz38CnQKc4SEupm4PyP5TmLvK64TdfO
       D7sno/W5oI1KZdimfW2c4r/6waNzZSvicMOWhLYY621Nn6njBc8VNwoxWpzCXhKm70b8
       +D4YZMn/eU5DN8dvhTv/bNK21FfJqjp033U=</G>
    <Y>W7dOmH/vWqocVCiqaxj6soxVXfR8XpMdY2Zv4Amjr3n81geyOLb6IZ+l7MUbdp8529DQ
       zuoVTthVpB9X4JKCprZIzifOTM1PFflTBzjx7egJwJWAIVdWyiIPjke6Va+wuV2n4Rl/
       cgCvrXK5cTov5C/Bpaf6o+qrrDGFBLLZTF4=</Y>
   </DSAKeyValue>
  </KeyValue>
 </KeyInfo>
</Signature>
```

Dane źródłowe - wskazanie

5.1. *Struktura podpisu*

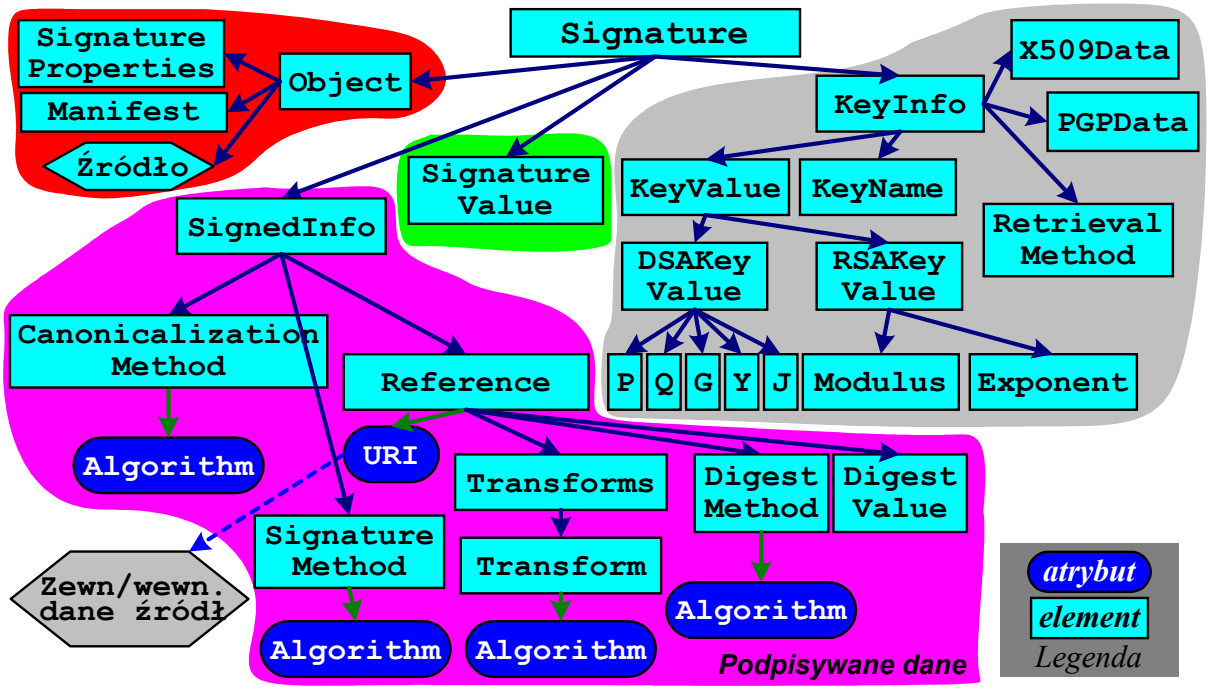
Element **`Signature`** posiada szereg elementów potomnych (rys. 3). Są to w szczególności:

- **`SignedInfo`** (obowiązkowy), zawiera informacje o podpisywanych danych poprzez podelementy:
  - **`CanonicalizationMehod`** – wymagany sposób sprowadzenia do postaci kanonicznej (uogólnionej), w tym np. sposób traktowania białych znaków (m.in. znaki nowej linii, wielokrotne spacje), omijanie komentarzy czy normalizacja kodowania znaków. Dotyczy to przekształcenia jedynie samego elementu **`SignedInfo`**.
  - **`SignatureMethod`** – wymagane określenie algorytmu stosowanego do generowania podpisu i sprawdzania jego poprawności, tj. funkcji kryptograficznych stosowanych przy tworzeniu skrótu, czy algorytmów asymetrycznych wykorzystywanych do szyfrowania skrótu. W standardzie wyróżniono

dwa podstawowe rodzaje podpisów: wymagany — DSS (jego implementacja jest konieczna, jeżeli aplikacja ma spełniać wymagania standardu) czyli DSA z SHA1 oraz zalecana RSA z SHA1.

- **Reference** – jedno lub wiele wskazań na podpisywane dane. Jest to przede wszystkim wskazanie na źródło (wyrażenie XPath i/lub adres URL) a także opis sposobu transformacji podpisu oraz metoda i wartość skrótu (niezaszyfrowana) — patrz dalej.

- **SignatureValue** (obowiązkowy) – wartość podpisu, tzn. zakodowana kluczem prywatnym nadawcy postać skrótu uzyskanego z elementu **SignedInfo**. Podpisywane są przede wszystkim wskazania na dane źródłowe oraz ich skróty zawarte w elemencie **Reference**. Wartość podpisu jest zawsze przekodowana do postaci tekstowej (z binarnej) za pomocą *Base64*.
- **KeyInfo** (opcjonalny) – informacja o kluczu publicznym niezbędnym do weryfikacji podpisu.
- **Object** (opcjonalny) – może zawierać podpisywane dane źródłowe, a także dodatkowe parametry podpisu (podelement **SignatureProperty**) takie, jak: format MIME, stempel czasowy, dane o zastosowanym urządzeniu kryptograficznym (np. numer fabryczny), a nawet zapisane w formacie XML instrukcje przetwarzania (odpowiednio interpretowane jedynie przez konkretną aplikację). W nim także można umieszczać wskazania na poszczególne fragmenty danych źródłowych, jeżeli podpisywane dane są złożeniem wielu części (podelement **Manifest** zawierający szereg odesłań — **Reference**).



Rys. 3. Elementy i atrybuty występujące w podpisie języka XML Signature

5.1.1. Informacja o danych źródłowych — element Reference

Element **Reference** zawiera atrybut **URI** który wskazuje na podpisywane dane źródłowe, umieszczone:

- w innym miejscu tego samego dokumentu XML, np. **URI="#DoPodpisu"** oznacza, że podpisywanmy jest element z tego samego dokumentu o atrybucie typu ID o wartości **DoPodpisu**; dozwolone są tutaj także inne wyrażenia XPath wskazujące na element dokumentu XML, np. **URI="//dane[@numer < 'K']/wartość"** — wskazanie elementu **wartość**, znajdującego się w elemencie **dane**, którego atrybut **@numer**, zaczyna się na literę mniejszą od „K", czyli od „A" do „J";
- w dokumencie dostępnym w sieci, np. **URI="http://przyklad.pl/dane/zrodla.xml"** określa dokument (najprawdopodobniej w formacie XML) dostępny na serwerze WWW **przyklad.pl** w pliku **zrodla.xml**, w kartotece **dane**;
- we fragmencie dokumentu dostępnego w sieci, np. **URI="http://przyklad.pl/zrodla.xml#DoPodpisu/1/3** oznacza trzeci podelement pierwszego dziecka (pierwszego podelementu) elementu o identyfikatorze **DoPodpisu**;
- w samym podpisie w innej jego części, wskazanie także poprzez wyrażenie XPath.

Oprócz tego w elemencie **Reference** istnieje inny atrybut, opisujący dodatkowo typ danych źródłowych: czy dane te zawarte są bezpośrednio w podpisie czy gdzieś na zewnątrz, wskazane w elemencie **Manifest**.

Jeżeli istnieje konieczność przetransformowania danych źródłowych, należy informacje o tym umieścić w podelemencie **Transform**. Dotyczy to w szczególności danych binarnych lub np. zakodowanych. W przypadku umieszczenia elementów **Transform** aplikacja przed podpisaniem przetwarza dane źródłowe za pomocą algorytmów wskazanych w kolejnych elementach **Transform**. Typowym sposobem przekształcania jest powszechnie znane kodowanie *Base64*.

Element **Reference** zawiera także informacje o algorytmie haszującym wykorzystanym do tworzenia skrótu (podelement **DigestMethod**). Zalecaną metodą jest SHA1, aczkolwiek możliwe jest zastosowanie dowolnej innej funkcji. W innym podelemencie (**DigestValue**) umieszcza się zakodowaną zawsze za pomocą *Base64*, niezabezpieczoną wartość skrótu.

5.1.2. Informacja o kluczu — element KeyInfo

Opcjonalny element **KeyInfo** zawiera dane o kluczu niezbędnym do weryfikacji podpisu, czyli potrzebnym do zdeszyfrowania skrótu. Informacja ta może być podana w bardzo różnej postaci. Dzięki rozszerzalności, która jest podstawową cechą języka XML, można stosować różne sposoby wskazania lub podania klucza wprost. W standardzie przewidziano przede wszystkim następujące rodzaje kluczy (odpowiednia wartość atrybutu **Type** elementu **RetrievalMethod**): DSA — rodzaj wymagany przez standard, RSA zalecany do zaimplementowania przez standard, certyfikat PKI w formacie X509, klucz w standardzie wykorzystywanym w PGP, SPKI a także podaną wprost i nie zalecaną — wartość klucza w postaci ciągu bajtów.

Oprócz tego w informacji o kluczu podawane są informacje umożliwiające jednoznaczne ustalenie klucza deszyfrującego u odbiorcy, m.in. poprzez elementy:

- **KeyName** — nazwa, identyfikator wskazujący jednoznacznie na klucz, np. identyfikator certyfikatu.
- **KeyValue** — wartość klucza publicznego podana wprost. Struktura tego elementu zależy ściśle od rodzaju klucza (przewidziano jedynie dwa rodzaje kluczy): dla RSA jest to **Modulus** (moduł dla funkcji modulo) oraz **Exponent** (wykładnik potęgi), zaś dla DSA są to: **P**, **Q**, **G**, **Y** ($Y=G^X$ mod P, gdzie X jest częścią klucza prywatnego), **J** ($J=(P-1)/Q$), **seed** (ziarno do generowania liczby pierwszej), **pgenCounter** (licznik do generowania liczby pierwszej).
- **X509Data** — dane opisujące jeden lub wiele certyfikatów X509, w szczególności:
  - nr seryjny certyfikatu nadany przez wystawcę — **X509IssuerSerial** (podobnie do **KeyName**),
  - dane o podmiocie na który wystawiono certyfikat (**X509SubjectName**),
  - dodatkowe informacje zawarte w rozszerzeniach (dotyczy certyfikatów w wersji 3) — **X509SKI**,
  - całą zawartość certyfikatu X509 — **X509Certificate**,
  - zawartość listy odwołanych certyfikatów CRL — **X509CRL**.
- **PGPData**. — dane opisujące klucz publiczny wg otwartego standardu PGP (*OpenPGP Message Format*)
- **RetrievalMethod** — wskazanie na magazyn z kluczem, umieszczony poza **KeyInfo**, w innym miejscu danego dokumentu lub gdzieś w sieci. Wskazanie takie jest to pomocne przy wielokrotnym wykorzystania tego samego klucza w wielu podpisach. Dzięki temu pozbywamy się zbędnej redundancji.

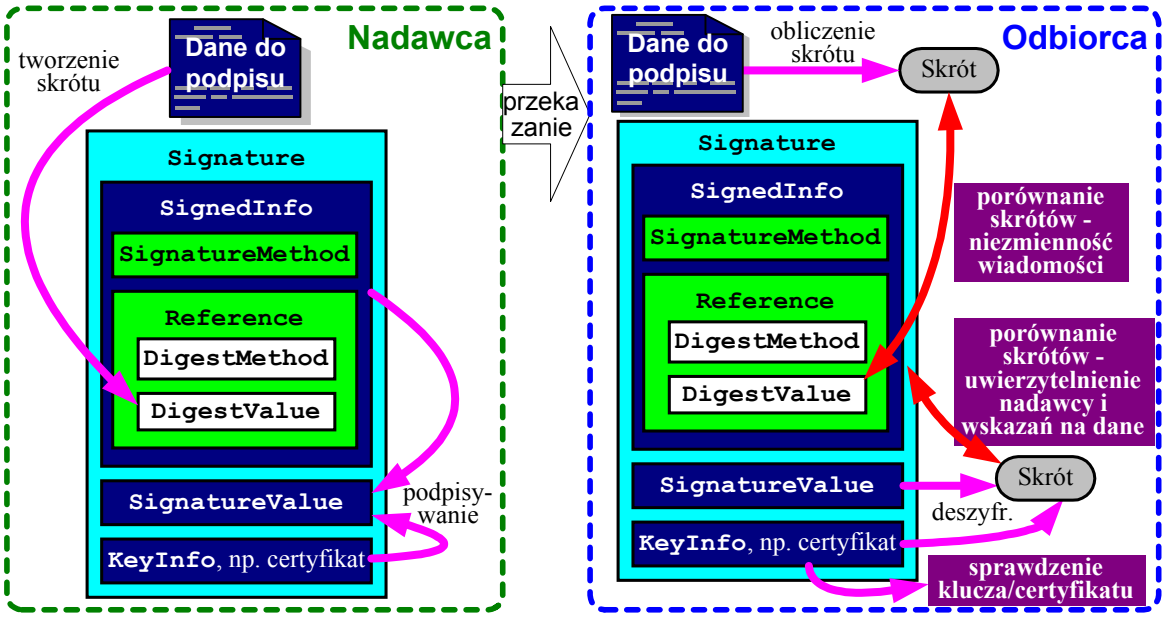
Duże liczby niezbędne w operacji deszyfrowania (np. **Modulus** dla RSA) są podawane w ściśle określony sposób — jako ciąg oktetów (bajtów) przekodowanych do postaci tekstowej za pomocą *Base64*.

5.2. Proces podpisywania i weryfikacji podpisu

Przy podpisywaniu aplikacja przetwarza dane źródłowe za pomocą transformacji wskazanych w elementach **Transform**. Z otrzymanego wyniku oblicza wartość skrótu i tworzy element **Reference** umożliwiający dotarcie do danych źródłowych, zawierający także (opcjonalnie) dane o sposobie transformacji, wskazanie na algorytm skrótu oraz wartość samego skrótu. Następnie tworzony jest element **SignedInfo**, po czym unifikowana (doprowadzana do postaci kanonicznej) jest jego postać. Element **SignedInfo**, jest następnie podpisywany (za pomocą właściwego klucza i algorytmu). Na końcu tworzona jest końcowa postać podpisu tj. element **Signature** zawierający wszystkie niezbędne (**SignedInfo**, **SignatureValue**) i ewentualnie opcjonalne wartości (np. element **KeyInfo** czy elementy **Object**).

Sprawdzanie podpisu (uwierzytelnianie podpisującego – rys. 4) następuje poprzez: uzyskanie — na podstawie danych zawartych w elemencie **KeyInfo** (może się to wiązać także z pobraniem go z zewnątrz) — klucza niezbędnego do deszyfracji. Następnie stosuje się algorytm podpisu (**SignatureMethod**) wraz z uzyskanym kluczem do zdekodowania podpisu (**SignatureValue**). Porównanie uzyskanej wartości ze skrótem przekształconego do postaci kanonicznej elementu **SignedInfo** (na podstawie **CanonicalizationMethod**) daje nam uwierzytelnienie nadawcy oraz „zaufane wskazania" na dane

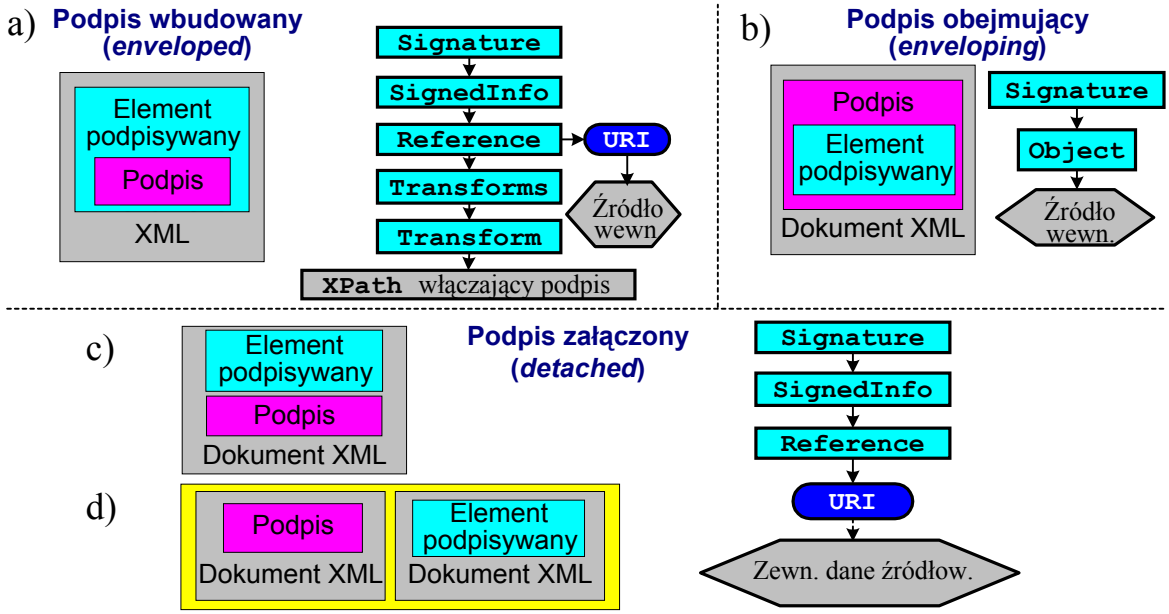
źródłowe. Sprawdzenie, czy wiadomość nie została zmieniona następuje poprzez obliczenie skrótu z wskazanej wiadomości (którą być może trzeba sprowadzić) w sposób analogiczny jak przy podpisywaniu i porównaniu z wartością skrótu z elementu **`DigestValue`**.



Rys. 4. Elementy i atrybuty występujące w podpisie języka XML Signature

5.3. Umiejscowienie podpisu

Podpis może być umieszczony w tym samym dokumencie XML, co podpisywane dane (rys. 5 a-c) lub w innym dokumencie (rys. 5d). Gdy podpis zawiera się w podpisywanym elemencie źródłowym — jest przezeń „obejmowany" (*enveloped*) – rys. 5a, wtedy w podpisie należy, oprócz wskazania na podpisywany element, umieścić także sposób usunięcia podpisu tak, aby móc otrzymać źródłowe dane bez podpisu. Można tego dokonać umieszczając odpowiednie wyrażenie XPath w transformacjach. Drugi sposób polega na objęciu podpisem danych źródłowych (rys. 5b), które umieszcza się w elemencie **`Object`**. Trzeci sposób — najbardziej typowy — to rozdzielenie podpisu i podpisywanych danych. Podpis i dane mogą być wtedy umieszczone w tym samym dokumencie (rys. 5c) lub w dwóch różnych dokumentach (rys. 5d). We wszystkich wymienionych przypadkach podpisywane dane źródłowe są wskazywane przez atrybut **`URI`** w elemencie **`Reference`**. W zależności od zastosowanego sposobu różna jest jedynie postać tego identyfikatora: odpowiednie wyrażenie XPath (w przypadku umieszczenia w tym samym dokumencie) lub adres URL (dla rozdzielonych dokumentów).



Rys. 5. Różne sposoby umiejscowienia podpisu względem podpisywanych danych

5.4. Inne standardy związane z podpisami w dokumentach XML. Implementacje

European Telecommunications Standards Institute (ETSI) zaproponował rozszerzenie XML Signature w postaci XML Advanced Electronic Signature (XAdES) [ETSI02a]. Uwzględnia on m.in. stemple czasowe oraz ułatwia realizację koncepcji podpisów długoterminowych. Polega to na dodawaniu, co jakiś czas — np. co dwa lata — dodatkowych podpisów cyfrowych wraz ze stemplami czasowymi, w których zastosowywane by były nowe algorytmy i ich parametry uznawane za bezpieczne w przyszłości.

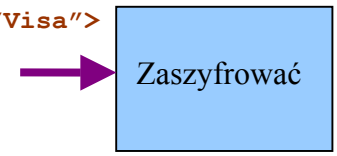
Lista wybranych implementacji standardu XML Signature jest dostępna m.in. pod adresem [Impl].

6. XML Encryption

Standard XML Encryption [Enc02] został zatwierdzony 10 grudnia 2002 r. i wraz ze standardem umożliwiający opisywanie wielokrotnych procesów deszyfrowania i weryfikacji podpisów powstałych w następstwie przetwarzania tego samego dokumentu (lub jego części) przez różne podmioty [Dec02]. Podobnie jak opisane wyżej podpisy, XML Encryption umożliwia szyfrowanie całego dokumentu XML, poszczególnych elementów lub ich zawartości (bez znaczników). Standard ten powstawał równolegle ze standardem podpisu elektronicznego XML Signature i ta koegzystencja ma swoje odzwierciedlenie w powiązaniu obu standardów. Dotyczy to w szczególności tego, że przy szyfrowaniu stosuje się informację o kluczu szyfrującym pochodzącą z przestrzeni nazw XML Signature.

Efektem szyfrowania jest element **`EncryptedData`**, zawierający zaszyfrowana postać danych wejściowych. W dokumencie może istnieć wiele elementów **`EncryptedData`** a także możliwe jest wielokrotne szyfrowanie tych samych danych, np. w jednym elemencie **`EncryptedData`** można umieścić zaszyfrowaną postać innego elementu **`EncryptedData`**. Weźmy przykładowy dokument XML zawierający informacje związane z płatnością, w tym m.in. numer karty kredytowej:

```
<?xml version='1.0'?>
<InfoPlatnicza xmlns='http://przyklad.pl/platnosc1'>
  <Nazwa>Józef Nowak</Nazwa>
  <KartaKredytowa Limit='2,000' Waluta='PLN' System="Visa">
    <NrKarty>4019 2445 0277 5567</NrKarty>
    <Wystawca>Nasz Bank S.A.</Wystawca>
    <DataWażnosci>10/03</DataWażnosci>
  </KartaKredytowa>
</InfoPlatnicza>
```



Oto wersja powyższego dokumentu, w której zaszyfrowano **cały element `KartaKredytowa`** (wraz z podelementami oraz znacznikami i atrybutami). Miejsce tego elementu zajął element **`EncryptedData`**:

```
<?xml version='1.0'?>
<InfoPlatnicza xmlns='http://przyklad.pl/platnosc1'>
  <Nazwa>Józef Nowak</Nazwa>
  <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
                 xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <CipherData>
      <CipherValue>A2s3B4f5gCbDyBreHwTWc5cx6weQ3g5tesV=</CipherValue>
    </CipherData>
 </EncryptedData>
</InfoPlatnicza>
```

Zauważmy, że w tak zabezpieczonym dokumencie postronny czytelnik nie wie nawet, jaki sposób płatności w nim zawarto. Gdyśmy jednak chcieli pozostawić niektóre informacje o sposobie płatności jawne, to można zaszyfrować jedynie **treść elementu `Nrkarty`** (bez znaczników i atrybutów):

```
<?xml version='1.0'?>
<InfoPlatnicza xmlns='http://przyklad.pl/platnosc1'>
 <Nazwa>Józef Nowak</Nazwa>
 <KartaKredytowa Limit='2,000' Waluta='PLN' System='Visa'>
  <NrKarty>
   <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
                  xmlns='http://www.w3.org/2001/04/xmlenc#'>
     <CipherData>
       <CipherValue>wKs1vFb2hBht3tCr5ex6dcfrDVgfGhRHhsJ4=</CipherValue>
     </CipherData>
   </EncryptedData>
  </NrKarty>
```

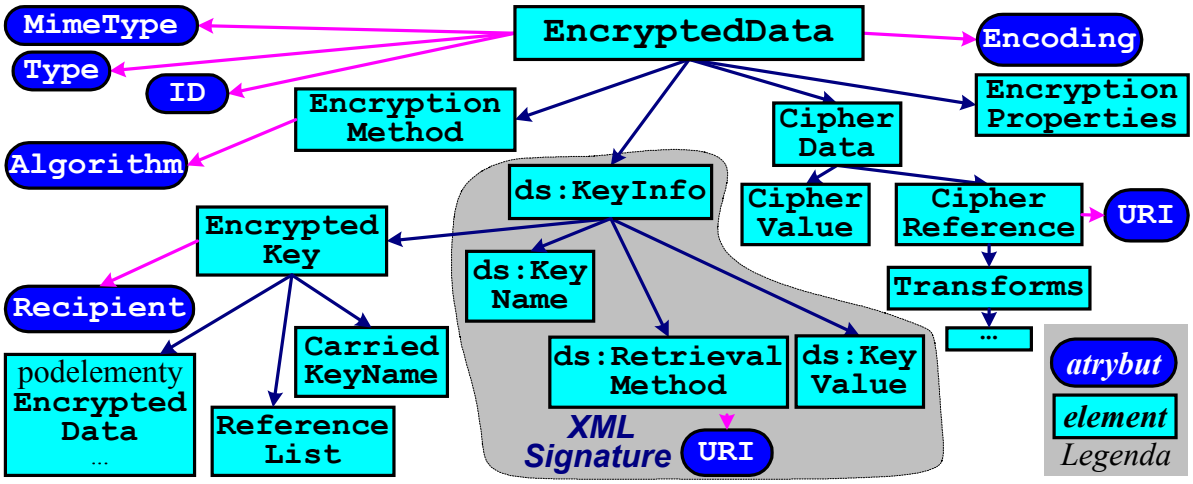
```
    <Wystawca>Nasz Bank S.A.</Wystawca>
    <DataWażnosci>10/03</DataWażnosci>
  </KartaKredytowa>
</InfoPlatnicza>
```

6.1. *Elementy XML Encryption*

Element główny zaszyfrowanych danych (**EncryptedData**) może zawierać atrybuty: **Id**, za pomocą którego można wskazywać na ten element, **MimeType** opisujący typ szyfrowanych danych oraz **Type** i **Encoding** opisujące odpowiednio typ szyfrowania i sposób przetwarzania danych wejściowych. Dodatkowo, element główny zawiera następujące podelementy (rys. 6):

- **CipherData**, w którym umieszcza się jedno z dwojga:
  - wprost zaszyfrowaną a następnie zakodowaną za pomocą *Base64* postać źródłową (**CipherValue**)
  - jedynie wskazanie na zaszyfrowana postać (**CipherReference**) wraz z ewentualnymi transformacjami, które powinny być dokonane przed deszyfrowaniem (jak w XML Signature)
- **EncryptionMethod** – zawierający wskazanie na metodę szyfrowania
- **EncryptionProperties** – wszelkie informacje dodatkowe o procesie szyfrowania takie jak np. stemple czasowe (czas dokonania szyfrowania), numer seryjny urządzenia szyfrującego
- **KeyInfo** – element pochodzący z przestrzeni nazw XML Signature, w którym umieszczono informacje o kluczu szyfrującym. Dodatkowym podelementem **KeyInfo** (nie przewidzianym przez XML Signature) może być zaszyfrowana postać klucza — element **EncryptedKey**, którego zawartość jest taka sama jak zawartość zwykłego szyfru (podelementy **EncryptedData**) z ewentualnie dodanymi: listą odesłań do zaszyfrowanych kluczy (**ReferenceList**), nazwą klucza czytelną dla człowieka (**CarriedKeyName**) oraz atrybutem **Recipient**, w którym można umieścić dowolne wskazanie np. nazwę odbiorcy zaszyfrowanej wiadomości.

Jak widać struktura informacji zawartej w szyfrze oferuje wiele możliwości, w tym np. szyfrowanie symetryczne połączone z przekazaniem klucza zaszyfrowanego asymetrycznym kluczem publicznym odbiorcy.



Rys. 6. Elementy XML Encryption

6.2. *Rodzaje stosowanych szyfrów*

Realizacja pełnego standardu XML Encryption powinna przewidywać obsługę następujących algorytmów:

- szyfry blokowe: potrójny DES – CBC, AES-128, AES-256 i opcjonalnie AES-192;
- dystrybucja kluczy symetrycznych: RSAES-PKCS1-v1_5 i RSAES-OAEP-ENCRYPT [PKCS1];
- do uzgadniania kluczy – opcjonalnie Diffie-Hellman;

a także algorytmy skrótów (SHA1 - wymagany, SHA256 - zalecany, SHA512, RIPEMD-160 - opcjonalne). Dopuszczalne jest również wykorzystanie dowolnych (własnych) algorytmów kryptograficznych.

7. Zaawansowane wykorzystanie standardów bezpieczeństwa

Oto kilka przykładów zaawansowanego wykorzystania standardów XML Signature i XML Encryption:

- Szyfrowanie zasadniczej treści kluczem symetrycznym oraz klucza symetrycznego asymetrycznym — typowe podejście w zabezpieczaniu informacji stosowane np. w PGP czy S/MIME.

- Szyfrowanie kluczy symetrycznych niezależnie wieloma kluczami publicznymi, co umożliwia przekazywanie informacji dla wielu odbiorców (wiele elementów **`EncryptedKey`**).
- Szyfrowanie różnych części dokumentu w różny sposób z przeznaczeniem dla różnych odbiorców, np. w XML-owym cenniku produktów element **`CenaProducenta`** dostępny jest jedynie dla oddziałów firmy, zaś **`CenaHurtowa`** dla dystrybutorów.
- Podpisywanie jednym podpisem wielu fragmentów pochodzących z jednego a nawet wielu dokumentów — podpisywana jest konkatenacja wszystkich fragmentów.
- Wiele podpisów w jednym dokumencie — wydzielenie (zebranie) podpisów z wielu dokumentów.
- Podpisywanie niezależnie przez wielu użytkowników, np. przez wszystkich członków zarządu.
- Wielokrotne podpisywanie i szyfrowanie dla zabezpieczenia całego złożonego procesu wymiany, np. szyfrowanie i podpisywanie danych o płatnościach w ofercie cenowej (sprzedawca) a następnie dodanie, zaszyfrowanych danych o numerze karty płatniczej i podpisanie całości przez klienta. Weryfikacja i niektóre dane dostępne byłyby jedynie dla banku — organizatora płatności. Pomocny jest wtedy standard opisu złożonego procesu deszyfrowania [Dec02].
- Podpisywanie / szyfrowanie danych binarnych w tym także multimedialnych (dźwięk, obrazy) i udostępnianie tekstowej, XML-owej wersji chronionych danych. Dokonuje się to poprzez transformacje danych binarnych do postaci tekstowej (elementy **`Transform`**).
- Umożliwienie tworzenia dokumentów zabezpieczonych długookresowo. Jest to związane z możliwością rozbudowywania podpisów i szyfrów o nowe elementy (cecha XML), np. [ETSI02a], a także tekstową a nie binarną postacią dokumentu XML, dostępną do przetwarzania w perspektywie dziesięcioleci.
- Wielokrotne szyfrowanie; dostęp do danych jedynie po zdeszyfrowaniu przez wszystkich odbiorców.

8. Problemy stosowania standardów bezpieczeństwa XML

Język XML powstał z myślą o systemie WWW, aczkolwiek obecnie obszary jego zastosowań zdecydowanie wykraczają poza ten system. Spójrzmy na pięć zagadnień wynikających głownie z charakterystyki języka XML, które mogą przysparzać pewnych problemów.

Pierwszy z problemów jest związany z **jednostkami zewnętrznymi** (*external entity*). Służą one do modularyzacji dokumentów XML — do jednego dokumentu można „włączać" inne, zewnętrzne dokumenty XML. Dzięki temu można np. budować katalogi produktów, w których opisy poszczególnych towarów są umieszczone w osobnych plikach. O tym, czy zewnętrzne jednostki mają być przetwarzane (włączane) czy nie, decyduje atrybut **`standalone`**, umieszczany w pierwszej linii dokumentu XML, tj. w deklaracji XML. Główny problem wynika z tego, że niektóre parsery (programy przetwarzające) nie obsługują jednostek zewnętrznych lub obsługują je wadliwie. Dodatkowo jednostki zewnętrzne mogą być niedostępne, np. umieszczone na serwerze, który został wyłączony. Jeżeli więc dokument XML zawierający odwołania do jednostek zewnętrznych zostanie zaszyfrowany i/lub podpisany, wtedy może pojawić się problem z prawidłowym zdeszyfrowaniem lub weryfikacją podpisu. Wniosek z tego taki, że w wymianie elektronicznej lepiej nie stosować jednostek zewnętrznych albo przynajmniej sprawdzić ich obsługę u wszystkich partnerów wymiany.

Kolejny problem wiąże się z **kodowaniem znaków**, w szczególności polskich liter [Kaz02e]. Otóż zalecanym standardem kodowania jest Unikod, jednakże dopuszcza się stosowanie innych standardów, np. Windows-1250, czy ISO Latin 2. Problem może wyniknąć np. przy szyfrowaniu lub podpisywaniu łącznie fragmentów wielu dokumentów XML, które są zakodowane w różnych standardach a fragmenty te nie zawierają w sobie deklaracji XML, czyli nie posiadają danych o sposobie kodowania. Przy deszyfracji odbiorca może błędnie odtworzyć lub zinterpretować uzyskane dane źródłowe (w złym standardzie).

Jeszcze inne zagadnienie jest związane z najczęstszym sposobem wykorzystania dokumentów XML w sieci Internet. Otóż są one zwykle **transformowane** do plików HTML (lub innej postaci) za pomocą opisów zawartych w osobnych dokumentach XSLT. Dokumenty HTML są następnie prezentowane użytkownikowi w przeglądarce. Poprzez dokumenty XSLT, które także są dokumentami XML, można dowolnie przekształcić dokument źródłowy, np. zwiększając dla pewnych warunków wszystkie występujące kwoty 10 razy. Odbiorca, jeżeli nie będzie miał gwarancji prawidłowości opisów transformacji, może być wprowadzony w błąd. Wprawdzie będzie on posiadał źródłowy dokument XML, ale nie będzie wiedział, że powinien go sprawdzić. W źródłowym dokumencie XML umieszcza się wskazanie (zawsze tylko jedno) — poprzez instrukcję przetwarzania **`xml-stylesheet`** — do dokumentu XSLT, opisującego transformację dokumentu. W związku z tym, jeżeli chcemy zabezpieczyć nie tylko dokument XML, ale także sposób jego prezentacji (lub ogólniej przetwarzania) to należy zastosować jeden z dwóch sposobów:

- Szyfrować / podpisywać zarówno dokument XSLT, jak i dokument XML (włącznie z instrukcją przetwarzania) — w szczególności można szyfrować / podpisywać samo wskazanie (instrukcję

przetwarzania). Dokument XSLT jest także dokumentem XML, więc można w nim wykorzystać ww. standardy bezpieczeństwa. Zauważmy, że w XML Signature można tworzyć podpisy dla złożenia kilku dokumentów XML, czyli dokumenty XML i XSLT można podpisać jednym podpisem.

- Wykorzystywać własne pliki XSLT odbiorcy, co jest typową praktyką, gdyż zwykle przesyłane są jedynie dane (źródłowy dokument XML) a nie sposoby jego prezentacji. Odbiorca wtedy albo sam tworzy pliki XSLT (co jest najczęściej spotykane) – ma więc do nich zaufanie, albo są one mu przekazane przez nadawcę wcześniej w bezpieczny sposób (np. także z zastosowaniem XML Encryption / Signature).

Jeżeli jednak nie zabezpieczymy odesłania ani nie zapewnimy bezpiecznego pliku z transformacjami nie możemy mieć zaufania do prezentowanych informacji. Dotyczy to także przekształceń dokonywanych w celu importu danych, np. do systemów firmowych lub wprost do baz Oracle czy MS SQL Server.

Kolejny problem to potencjalna możliwość ataków **DoS** (*Denial od Service*) na systemy odbiorcy, który deszyfruje zabezpieczone dokumenty XML. Polegają one na „zalaniu pracą" systemu deszyfrujacego, lub jego „zapętleniu", które jest skutkiem możliwej rekurencji. Zagrożenie to wynika z dużej elastyczności standardów XML, w tym także XML Encryption. Jeżeli do deszyfracji potrzebujemy zdeszyfrowanego klucza A (element **`EncryptedKey`**), który wymaga zdeszyfrowania klucza B (inny element **`EncryptedKey`**), to jeżeli klucz B korzysta z klucza A — mamy rekurencję. Inny scenariusz polega na umieszczeniu w podpisach lub szyfrach odesłań do zasobów sieciowych, które są bardzo duże (mocno obciążają zasoby odbiorcy) lub wielokrotnie odsyłają dalej, do kolejnych zasobów.

Ostatnie zagrożenie jest podobne jak dla wszystkich metod kryptograficznych. Przekazywanie zaszyfrowanych dokumentów może skutkować przesyłaniem danych, które mogą stanowić zagrożenie. Systemy zabezpieczeń — np. zapory ogniowe, systemy wykrywania włamań — nie są w stanie kontrolować zaszyfrowanych treści. Analogiczny problem dotyczy jednak także innych bezpiecznych systemów takich, jak: SSL, VPN, S/MIME czy PGP.

9. Podsumowanie

Standardy bezpieczeństwa XML Signature i XML Encryption umożliwiają bardzo elastyczną ochronę dokumentów XML. Ponieważ język XML znajduje zastosowanie także dla opisu niektórych danych multimedialnych (rys. 2), więc w prosty sposób można zabezpieczać za pomocą jego mechanizmów grafiki wektorowe SVG, dokumenty rzeczywistości wirtualnej VRML (X3D jest XML-owym następcą języka VRML), pliki XSL-FO, z których generowane są dokumenty PDF, serwisy WAP czy wzory matematyczne.

Oprócz tego, zarówno w podpisach XML Signature jak i szyfrach XML Encryption dostępne są transformacje danych binarnych do tekstu, co daje możliwość ochrony dowolnych danych binarnych. Największym atutem prezentowanych mechanizmów jest ich otwartość, bezpłatność a przede wszystkim integracja z językiem XML, który jest powszechnie stosowany w systemach informacyjnych do wymiany danych. Dodatkowo, tak zaszyfrowane dane lub podpisy mogą być rozbudowywane i przechowywane przez wiele lat. Za pomocą XML Signature oraz XML Encryption można zabezpieczać niemal dowolne fragmenty dokumentów XML, co może mieć istotne znaczenie dla biznesowych procesów wymiany danych (EDI XML).

Literatura

[Dec02] Hughes M., Imamura T., Maruyama H. (eds): *Decryption Transform for XML Signature. W3C Recommendation 10 December 2002*. WWW Consortium, http://www.w3.org/TR/xmlenc-decrypt

[Enc02] Imamura T., Dillaway B., Simon E.: *XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002*. WWW Consortium, http://www.w3.org/TR/xmlenc-core/

[ETSI02a] *XML Advanced Electronic Signatures (XAdES). ETSI TS 101 903 V1.1.1*. European Telecommunications Standards Institute, 2002-02, http://pda.etsi.org/pda/home.asp?wki_id=12532.

[ETSI02b] *TC Security – Electronic Signatures and Infrastructures (ESI); XML format for signatures policies. ETSI TR 102 038 v.1.1.1*. European Telecommunications Standards Institute ETSI, 2002-04.

[Impl] *XML Signature Developer Resources*. http://www.xmltrustcenter.org/xmlsig/developer/index.htm

[Kaz02a] Kazienko P.: *Bezpieczeństwo dokumentów XML czyli XML Signature i XML Encryption*. W: XML w praktyce. Software Konferencje, Warszawa, 6-7 listopad 2002.

[Kaz02b] Kazienko P.: *Modelowanie dokumentów XML Schema*. W: XML w praktyce. Materiały konferencyjne. Software Konferencje, Warszawa, 6-7 listopad 2002, s. 29-44.

[Kaz02c] Kazienko P.: *Ochrona selektywna.* Computerworld Raport, grudzień 2002, s. 24-25.

[Kaz02d] Kazienko P.: *Rodzina języków XML.* Software 2.0 nr 6 (90), czerwiec 2002, s. 22-27.

[Kaz02e] Kazienko P.: *XML a sprawa polska*. Chip Special. Poradnik Webmastera, nr 11(69) 2002, s. 69-71.

[Kaz02f] Kazienko P., Gwiazda K.: *XML na poważnie*. Helion, Gliwice, 2002.

[PKCS1] Kaliski B., Staddon J.: *PKCS #1: RSA Cryptography Specifications. Version 2.0. RFC 2437*. Network Working Group, October 1998, http://www.ietf.org/rfc/rfc2437.txt.

[Sig02] Bartel M., Boyer J., Fox B., LaMacchia B., Simon E.: *XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002*. WWW Consortium, http://www.w3.org/TR/xmldsig-core/