ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2012
Seria: ORGANIZACJA I ZARZĄDZANIE z. 61 Nr kol. 1873

Ireneusz J. JÓŹWIAK, Grzegorz M. KOZŁOWSKI
Politechnika Wrocławska
Wydział Informatyki i Zarządzania
Instytut Informatyki

PRZEGLĄD SYSTEMÓW DO PODPISYWANIA KODU

Streszczenie. W artykule przedstawiono wiodące rozwiązania z dziedziny systemów do podpisywania kodu aplikacji. Systemy Microsoft Authenticode, Oracle Java Code Signing oraz Apple Codesign zostały porównane pod kątem: ich założeń projektowych, struktury wytwarzanego podpisu cyfrowego, a także sposobów jego generowania oraz weryfikacji. Została również omówiona koncepcja tożsamości kodu źródłowego aplikacji.

REVIEW OF CODE SIGNING SYSTEMS

Summary. This publication discusses the leading solutions for code signing systems such as Microsoft Authenticode, Oracle Java Code Signing and Apple Codesign. All systems were compared to each other in aspects such as digital signature structure, its generation and verification. A special attention was given to an issue of code identity.

1. Wprowadzenie

Podpisanie kodu źródłowego aplikacji jest procesem cyfrowego podpisywania plików wykonywalnych oraz skryptów w celach potwierdzenia tożsamości autora oprogramowania oraz zagwarantowania, że kod źródłowy nie został zmieniony lub też uszkodzony podczas jego dystrybucji. Podpisywanie kodu źródłowego aplikacji jest szczególnie istotne w przypadku dystrybucji oprogramowania za pomocą niezabezpieczonych bądź też słabo zabezpieczonych kanałów dystrybucji, w których kod źródłowy całości lub wyłącznie danego fragmentu aplikacji może zostać w każdej chwili wyświetlony przez osoby postronne.

Przykładem takiej sytuacji może być dystrybucja oprogramowania w formie apletów Javy, kontrolek ActiveX bądź też skryptów wykonywanych przez przeglądarki internetowe. Innym ważnym sposobem wykorzystania systemów do podpisywania kodu źródłowego aplikacji jest bezpieczne dostarczenie aktualizacji i poprawek do już zainstalowanego oprogramowania u klienta. Większość dystrybucji systemu operacyjnego Linux, systemy operacyjne Apple Mac OS X oraz Microsoft Windows wykorzystują usługi systemów podpisywania kodu źródłowego w celu zapewnienia, że dostarczane aktualizacje w żaden sposób nie zostały zmodyfikowane podczas ich dystrybucji do klienta. Pozwala to na zaniechanie troski o bezpieczeństwo kanałów dystrybucyjnych oprogramowania, zwłaszcza w przypadku stosowania takich rozwiązań, jak na przykład serwery lustrzane, mogące nie znajdować się pod pełną kontrolą autorów dystrybuowanego oprogramowania.

Budowa systemu do podpisywania kodu źródłowego aplikacji bazuje na szeroko omawianej w literaturze ([1], [4], [5] i [6]) infrastrukturze klucza publicznego, a także na koncepcji podpisu cyfrowego. Jednak podpis ten ma zastosowanie w odniesieniu do dystrybuowanego oprogramowania komputerowego, a nie dokumentów. Budowa przykładowego systemu podpisu cyfrowego opiera się na kryptograficznych algorytmach asymetrycznych, takich jak algorytm: Digital Signature Standard, Rivesta, Shamira i Adlemana czy ElGamal. Dodatkowo wykorzystywane są algorytmy jednokierunkowych funkcji skrótu, takie jak algorytmy z rodzin Message-Digest, Secure Hash Algorithm czy N-Hash. Dokładne omówienie matematyczne tych algorytmów można znaleźć w literaturze [9].

Wśród przykładów udanych implementacji systemów do podpisywania kodu źródłowego aplikacji można wymienić systemy: Microsoft Authenticode, Apple Codesign oraz Oracle Java Code Signing. Pierwszy z nich został zaprojektowany w celach generowania oraz weryfikacji podpisów cyfrowych aplikacji, zaprojektowanych na podstawie platformy Microsoft .NET. System Apple Codesign, będący częścią zintegrowanego środowiska programistycznego Apple Xcode, służy ochronie aplikacji napisanych w językach C, C++, Objective-C oraz Objective-C++, przeznaczonych dla systemów operacyjnych wyprodukowanych przez firmę Apple. Ostatni z wymienionych systemów do podpisywania kodu źródłowego aplikacji – system Oracle Java Code Signing – w głównej mierze jest przeznaczony do ochrony apletów internetowych, zaprojektowanych na podstawie platformy Oracle Java [10].

Celem artykułu jest dokonanie przeglądu istniejących rozwiązań z zakresu systemów do podpisywania kodu źródłowego aplikacji. Zaprezentowane zostaną systemy: Microsoft Authenticode, Apple Codesign oraz Oracle Java Code Signing. Zostaną one porównane pod kątem struktury oraz założeń projektowych, a także pod kątem sposobu generowania i weryfikacji podpisów cyfrowych aplikacji.

2. Tożsamość kodu

Tożsamość kodu jest definiowana przez określenie związku pomiędzy podpisanym cyfrowo kodem aplikacji a osobą fizyczną bądź też organizacją, która wykonała dany podpis cyfrowy i jest gwarantem jego autentyczności. Dobry przykład może stanowić sytuacja, w której instytucja będąca producentem oprogramowania ma certyfikat cyfrowy, wydany i poświadczony przez urząd rejestracji. Certyfikat ten służy producentowi oprogramowania do generowania podpisów cyfrowych wszystkich aplikacji wytwarzanych przez tę firmę. Tym samym każda podpisana w ten sposób aplikacja ma poświadczoną przez jej producenta tożsamość kodu, co pozwala na jej jednoznaczną identyfikację i powiązanie z tożsamością jej wytwórcy. Dodatkowo aplikacje wytwarzane przez danego producenta oprogramowania tworzą swego rodzaju strukturę drzewiastą, w której każdy produkt ma tę samą tożsamość kodu bez względu na jego aktualną wersję.

W niektórych sytuacjach określenie tożsamości kodu może stanowić wynikową wielu różnych jego tożsamości. Sytuacja taka ma miejsce w przypadku podpisów modułowych, w których tożsamość kodu danej aplikacji jest określana na podstawie tożsamości danego kodu oraz tożsamości wszystkich powiązanych z nim bibliotek. Jest to jednak możliwe wyłącznie wówczas, gdy biblioteki powiązane z daną aplikacją zostały uprzednio podpisane cyfrowo przez ich producentów.

Każda nieautoryzowana zmiana w podpisanym cyfrowo kodzie aplikacji natychmiast powoduje utratę tożsamości kodu. Aplikacja taka jest wówczas traktowana przez system do podpisywania kodu źródłowego na równi z innymi aplikacjami całkowicie pozbawionymi podpisu cyfrowego.

3. Założenia projektowe

Budowa systemu do podpisywania kodu źródłowego Microsoft Authenticode została opracowana z myślą o tworzeniu podpisów cyfrowych aplikacji dystrybuowanych w ramach systemów operacyjnych z rodziny Microsoft Windows, opartych na platformie Microsoft.NET [8]. W przypadku systemów Oracle Java Code Signing i Apple Codesign platformami tymi są odpowiednio Oracle Java oraz Apple Cocoa. Wszystkie systemy do podpisywania kodu źródłowego aplikacji zostały zaprojektowane tak, aby ich działanie w maksymalnym stopniu było przystosowane do pracy z dedykowanymi im, zintegrowanymi środowiskami programistycznymi.

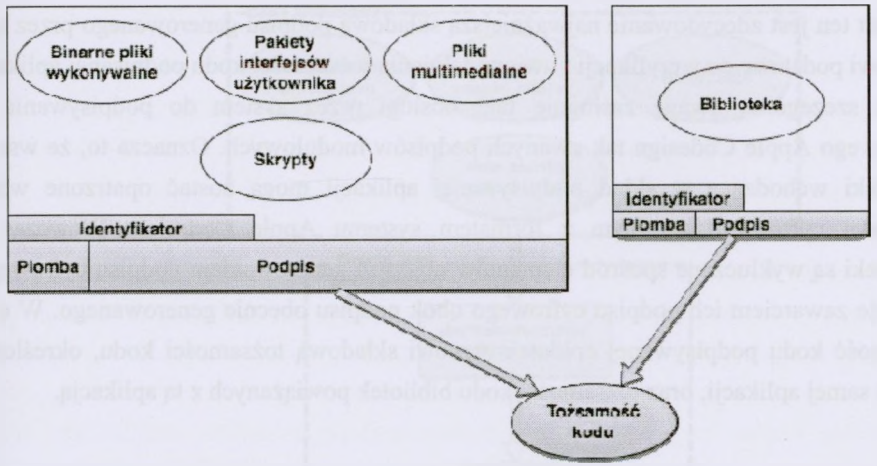
Głównym założeniem wszystkich omawianych systemów do podpisywania kodu źródłowego jest możliwość cyfrowego podpisywania wszystkich wykonywalnych zasobów

i skryptów wchodzących w skład podpisywanej aplikacji. Wśród tych plików należy wyróżnić zarówno pliki wykonywalne, skrypty, pakiety interfejsów użytkownika, biblioteki, archiwa skompresowane, jak i pliki multimedialne, takie jak pliki dźwiękowe bądź graficzne. Dodatkowo w przypadku systemu Microsoft Authenticode podpisem cyfrowym objęte są kontrolki ActiveX [8]. Wart podkreślenia jest fakt, iż w przypadku systemu Apple Codesign osobnym podpisem cyfrowym mogą zostać objęte biblioteki programistyczne, dystrybuowane wraz z podpisywaną aplikacją. Jest to efektem obsługi przez ten system podpisu modułowego, pozwalającego na wykorzystanie już istniejących i pomyślnie zweryfikowanych podpisów cyfrowych, którymi zostały opatrzone fragmenty podpisywanej aplikacji jako części ostatecznego podpisu wynikowego, generowanego przez system dla danej aplikacji.

Bardzo istotnym założeniem wszystkich omawianych systemów do podpisywania kodu źródłowego aplikacji jest możliwość wykonywania weryfikacji podpisu cyfrowego oprogramowania przez przeprowadzanie tak zwanej weryfikacji statycznej, a w przypadku systemów Microsoft Authenticode oraz Apple Codesign również weryfikacji dynamicznej. Oznacza to możliwość wykonywania weryfikacji podpisu cyfrowego przez system do podpisywania kodu źródłowego zarówno w przypadku aplikacji uruchomionych w systemie operacyjnym, jak i aplikacji nieuruchomionych. W praktyce daje to możliwość weryfikacji podpisu cyfrowego oprogramowania w momencie jego instalacji bądź też dystrybucji, co jest możliwe dzięki przeprowadzeniu weryfikacji statycznej podpisu. Aplikacje uruchomione w systemie operacyjnym mogą zostać poddane weryfikacji w celu sprawdzenia, czy nie uległy one modyfikacji od momentu ich uruchomienia, bądź też w celu autoryzacji ich dostępu do zasobów systemu operacyjnego. Przeprowadzenie weryfikacji dynamicznej podpisu cyfrowego jest możliwe dzięki silnej integracji systemu do podpisywania kodu źródłowego z odpowiadającą mu platformą programistyczną.

4. Struktura podpisu

Struktura podpisu cyfrowego, generowanego przez wszystkie omawiane systemy do podpisywania kodu źródłowego, wynika wprost z założeń postawionych podczas procesu ich projektowania. Fizyczną reprezentację podpisu cyfrowego stanowi plik tekstowy zawierający wynik procesu podpisywania kodu, wygenerowany przez system do podpisywania kodu źródłowego. Plik ten jest dystrybuowany razem z gotową aplikacją i stanowi jej integralną część. Szczegółowa struktura podpisu cyfrowego generowanego przez system Apple Codesign została zaprezentowana na rysunku 1.



Rys. 1. Struktura podpisu cyfrowego systemu do podpisywania kodu źródłowego Apple Codesign [2]
Fig. 1. Digital signature structure in Apple Codesign system [2]

Każdy podpis cyfrowy, wygenerowany przez omawiane systemy, zawiera w swojej strukturze unikalny identyfikator przypisany danej aplikacji przez system do podpisywania kodu źródłowego. Identyfikator ten pozwala na jednoznaczne rozpoznawanie przez system podpisanych aplikacji i powiązanie ich z określonymi tożsamościami kodu. Dotyczy to zarówno identyfikacji wytwórcy danego oprogramowania, jak i samej aplikacji oraz jej wersji. Tak skonstruowany identyfikator pozwala na określenie uprawnień danej aplikacji na podstawie reguł przypisanych danemu wydawcy oprogramowania bądź też danej rodzinie aplikacji.

Bardzo istotnym elementem składowym, wchodzącym w skład podpisu cyfrowego wygenerowanego przez każdy z omawianych systemów do podpisywania kodu źródłowego aplikacji, jest tak zwana plomba. Przez to pojęcie należy rozumieć skrót całości podpisanej aplikacji, wygenerowany przez jednokierunkową funkcję skrótu wchodzącą w skład systemu. Skrót ten jest generowany na podstawie wszystkich plików objętych podpisem cyfrowym aplikacji i jest zapisywany w formie tekstowej, reprezentowanej przez liczbę w systemie szesnastkowym. Plomba aplikacji stanowi podstawę do wygenerowania podpisu cyfrowego aplikacji przez algorytm asymetryczny, wchodzący w skład systemu do podpisywania kodu źródłowego aplikacji. Element ten jest gwarantem niezmienności danych objętych podpisem cyfrowym i stanowi podstawę do weryfikacji podpisu cyfrowego.

Ostatnim obowiązkowym elementem zawartym w strukturze podpisu aplikacji jest sam podpis cyfrowy. Jest on generowany przez algorytm asymetryczny, wchodzący w skład systemu do podpisywania kodu źródłowego na podstawie plomby aplikacji, a następnie

zapisywany w formie tekstowej, reprezentowanej przez liczbę w systemie szesnastkowym. Element ten jest zdecydowanie najważniejszą składową podpisu generowanego przez system i stanowi podstawę do weryfikacji i uwierzytelnienia tożsamości kodu podpisanej aplikacji.

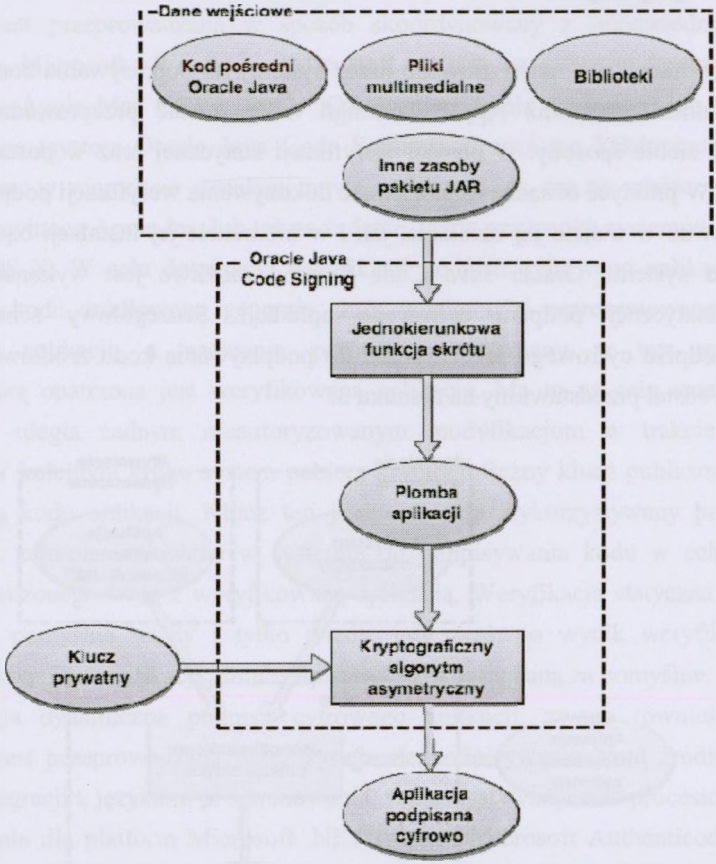
Na szczególną uwagę zasługuje fakt obsługi przez system do podpisywania kodu źródłowego Apple Codesign tak zwanych podpisów modułowych. Oznacza to, że wszystkie biblioteki wchodzące w skład podpisywanej aplikacji mogą zostać opatrzone własnym podpisem cyfrowym, zgodnym z formatem systemu Apple Codesign. Wówczas takie biblioteki są wykluczane spośród elementów objętych generowaniem podpisu cyfrowego, co skutkuje zawarciem ich podpisu cyfrowego obok podpisu obecnie generowanego. W efekcie tożsamość kodu podpisywanej aplikacji stanowi składową tożsamości kodu, określonej na użytek samej aplikacji, oraz tożsamości kodu bibliotek powiązanych z tą aplikacją.

5. Proces podpisywania kodu

Proces generowania podpisu cyfrowego przez każdy z omawianych systemów do podpisywania kodu źródłowego odbywa się w trzech etapach i w przypadku systemów Microsoft Authenticode oraz Apple Codesign jest ściśle powiązany z narzędziami dostarczanymi programistom przez zintegrowane środowiska programistyczne Microsoft Visual Studio oraz Apple Xcode. Schemat procesu podpisywania kodu przez system Oracle Java Code Signing został zaprezentowany na rysunku 2.

Pierwszym etapem procesu podpisywania kodu jest dostarczenie danych wejściowych w postaci plików objętych ochroną systemu do podpisywania kodu źródłowego aplikacji. Na podstawie tych danych w kolejnych etapach zostanie wygenerowany podpis cyfrowy, aplikacji. W tym etapie do systemu jest również dostarczany certyfikat cyfrowy, jednoznacznie identyfikujący tożsamość osoby, bądź też certyfikat instytucji odpowiedzialnej za poświadczenie tożsamości kodu podpisywanej aplikacji. Certyfikat w standardzie X.509 może zostać dostarczony do systemu do podpisywania kodu źródłowego w dwóch możliwych wariantach: certyfikat poświadczony lub niepoświadczony przez urząd certyfikacji. Bez względu na to, w jakiej formie certyfikat zostanie dostarczony, stanie się on podstawą do wygenerowania podpisu cyfrowego aplikacji.

W kolejnym etapie system do podpisywania kodu źródłowego generuje tak zwaną plombę na podstawie danych wejściowych dostarczonych do systemu w poprzednim etapie. W tym celu wykorzystywana jest jednokierunkowa funkcja skrótu, zaimplementowana w systemie. Generuje ona skrót wiadomości reprezentowanej przez dane podpisywanej aplikacji. Skrót ten stanowi podstawę do wygenerowania podpisu cyfrowego, a także zabezpiecza aplikację przed dokonaniem w niej nieautoryzowanych zmian w przyszłości.

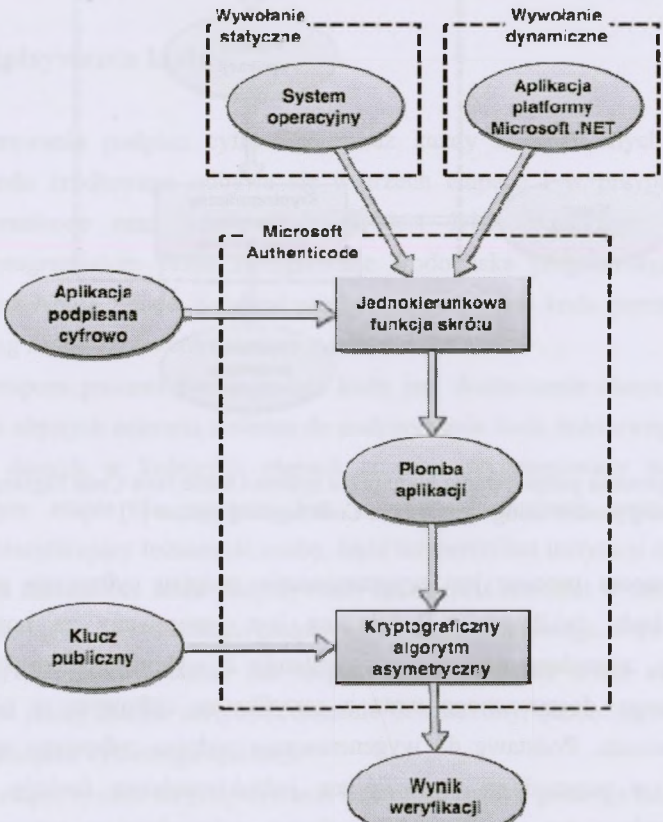


Rys. 2. Schemat procesu podpisywania kodu przez system Oracle Java Code Signing [7]
Fig. 2. Code signing process using Oracle Java Code Signing system [7]

Ostatnim etapem procesu jest wygenerowanie podpisu cyfrowego przez system do podpisywania kodu źródłowego. Podpis ten jest generowany za pomocą algorytmu asymetrycznego, zaimplementowanego w systemie z wykorzystaniem kryptograficznego klucza prywatnego, dostarczonego wraz z certyfikatem cyfrowym w pierwszym etapie omawianego procesu. Podstawę do wygenerowania podpisu cyfrowego aplikacji stanowi, wygenerowana w poprzednim etapie przez jednokierunkową funkcję skrótu, plomba aplikacji. Powstały w ten sposób podpis cyfrowy, poświadczający tożsamość kodu danej aplikacji, jest zapisywany wraz z plombą aplikacji w postaci tekstowej, reprezentowanej w systemie szesnastkowym.

6. Weryfikacja podpisu

Proces weryfikacji podpisu cyfrowego przez systemy do podpisywania kodu źródłowego Microsoft Authenticode oraz Apple Codesign może zostać przeprowadzony na dwa, niezależne od siebie sposoby: w postaci weryfikacji statycznej oraz w postaci weryfikacji dynamicznej. W praktyce oznacza to możliwość dokonywania weryfikacji podpisu cyfrowego aplikacji zarówno w trakcie jej działania, jak i w momencie jej instalacji bądź dystrybucji. W przypadku systemu Oracle Java Code Signing możliwe jest wykonanie wyłącznie weryfikacji statycznej podpisu cyfrowego aplikacji. Szczegółowy schemat procesu weryfikacji podpisu cyfrowego przez system do podpisywania kodu źródłowego Microsoft Authenticode został przedstawiony na rysunku 3.



Rys. 3. Schemat procesu weryfikacji podpisu cyfrowego aplikacji przez system Microsoft Authenticode [3]

Fig. 3. Digital signature verification process using Microsoft Authenticode system [3]

Weryfikacja statyczna podpisu cyfrowego aplikacji, zwana również weryfikacją zewnętrzną, jest przeprowadzana w sposób skoordynowany z odpowiednim systemem operacyjnym: Microsoft Windows, Microsoft Windows Phone (system Microsoft Authenticode), Apple Mac OS X, Apple iOS (system Apple Codesign) lub też platformą programistyczną (system Oracle Java Code Signing). Proces ten każdorazowo może być przeprowadzany w momencie instalacji nowej aplikacji, jak ma to miejsce w przypadku systemu operacyjnego Apple Ios, lub też na żądanie, jak w przypadku systemów operacyjnych Apple Mac OS X. W celu dokonania weryfikacji podpisu cyfrowego aplikacji system do podpisywania kodu źródłowego generuje skrót wiadomości reprezentowanej przez dane podpisywanej aplikacji, a następnie porównuje otrzymany w ten sposób wynik z plombą, którą opatrzona jest weryfikowana aplikacja. Ma to na celu sprawdzenie, czy aplikacja nie uległa żadnym nieautoryzowanym modyfikacjom w trakcie procesu jej dystrybucji. W kolejnym kroku system pobiera kryptograficzny klucz publiczny, powiązany z tożsamością kodu aplikacji. Klucz ten jest następnie wykorzystywany przez algorytm asymetryczny, zaimplementowany w systemie do podpisywania kodu w celu weryfikacji podpisu dostarczonego wraz z weryfikowaną aplikacją. Weryfikacja statyczna aplikacji jest uznawana za pomyślną wtedy i tylko wtedy, gdy zarówno wynik weryfikacji podpisu aplikacji, jak i wynik weryfikacji plomby aplikacji uznane zostaną za pomyślne.

Weryfikacja dynamiczna podpisu cyfrowego aplikacji, zwana również weryfikacją wewnętrzną, jest przeprowadzana przez system do podpisywania kodu źródłowego dzięki jego silnej integracji z językami programowania, wykorzystywanymi w procesie wytwarzania oprogramowania dla platform Microsoft .NET (system Microsoft Authenticode) lub Apple Cocoa (system Apple Codesign). Proces ten jest przeprowadzany na działającej w systemie operacyjnym aplikacji na jej własne żądanie lub też na żądanie innej działającej aplikacji. W celu dokonania weryfikacji podpisu cyfrowego aplikacji system do podpisywania kodu źródłowego generuje skrót wiadomości reprezentowanej przez dane podpisywanej aplikacji, a następnie porównuje otrzymany w ten sposób wynik z plombą, którą opatrzona jest weryfikowana aplikacja. Ma to na celu sprawdzenie, czy aplikacja nie uległa żadnym nieautoryzowanym modyfikacjami od momentu jej uruchomienia lub też w trakcie procesu jej dystrybucji. W kolejnym kroku system pobiera kryptograficzny klucz publiczny, powiązany z tożsamością kodu aplikacji. Klucz ten jest następnie wykorzystywany przez algorytm asymetryczny, zaimplementowany w systemie do podpisywania kodu w celu weryfikacji podpisu dostarczonego wraz z weryfikowaną aplikacją. Weryfikacja dynamiczna aplikacji jest uznawana za pomyślną wtedy i tylko wtedy, gdy zarówno wynik weryfikacji podpisu aplikacji, jak i wynik weryfikacji plomby aplikacji uznane zostaną za pomyślne.

7. Podsumowanie

Przeprowadzona analiza trzech systemów do podpisywania kodu źródłowego aplikacji pozwala na wskazanie zarówno istotnych różnic, jak i cech wspólnych wszystkich omówionych systemów. Wszystkie spośród trzech przeanalizowanych systemów do podpisywania kodu źródłowego aplikacji cechuje integracja bądź to z obsługiwaną przez dany system platformą programistyczną, bądź z obsługiwanymi systemami operacyjnymi. Tylko dwa spośród przeanalizowanych rozwiązań cechują się silną integracją ze zintegrowanymi środowiskami programistycznymi. Jest to spowodowane faktem, iż zarówno system Microsoft Authenticode, jak i system Apple Codesign stanowią część składową platform programistycznych, dysponujących dedykowanymi im zintegrowanymi środowiskami programistycznymi. Wszystkie przeanalizowane systemy do podpisywania kodu źródłowego cechuje możliwość cyfrowego podpisywania wszystkich zasobów wchodzących w skład podpisywanej aplikacji. Jednak wyłącznie system Apple Codesign umożliwia wykonanie tak zwanego podpisu modułowego, pozwalającego na określenie tożsamości podpisanego kodu jako wynikowej tożsamości tego kodu i kodu bibliotek powiązanych z daną aplikacją. Istotny jest również fakt, iż zarówno system Microsoft Authenticode, jak i system Apple Codesign stanowią obowiązujący standard przy wytwarzaniu aplikacji na dedykowane im urządzenia mobilne, co przejawia się w wymogu wykonania podpisu cyfrowego każdej aplikacji dystrybuowanej dla systemu operacyjnego Apple iOS oraz nowych wersji systemu operacyjnego Microsoft Windows Phone.

Budowa wszystkich trzech systemów do podpisywania kodu źródłowego aplikacji bazuje na wspomnianej już infrastrukturze klucza publicznego oraz na dostarczanej przez nią funkcjonalności generowania podpisu cyfrowego dokumentów. Oznacza to, że każdy z przeanalizowanych tutaj systemów do podpisywania kodu źródłowego aplikacji opiera swoją budowę na zaimplementowanym kryptograficznym algorytmie asymetrycznym oraz wybranym algorytmie jednokierunkowej funkcji skrótu. Wszystkie z przeanalizowanych systemów do podpisywania kodu źródłowego aplikacji w trakcie generowania podpisu cyfrowego tworzą tak zwaną plombę aplikacji, zabezpieczającą ją przed nieautoryzowanymi modyfikacjami podczas procesu jej dystrybucji. Również każdy z systemów opiera wykonanie podpisu cyfrowego na uprzednio wygenerowanej plombie aplikacji. Wszystkie systemy zapisują zarówno podpis cyfrowy, jak i plombę aplikacji w postaci tekstowej, reprezentującej wygenerowane dane w systemie szesnastkowym.

Każdy z przeanalizowanych systemów do podpisywania kodu źródłowego aplikacji umożliwia wykonanie tak zwanej weryfikacji statycznej podpisu cyfrowego aplikacji. Oznacza to, że każdy z systemów pozwala na wykonanie weryfikacji tożsamości kodu

nieuruchomionej jeszcze aplikacji zarówno podczas procesów jej dystrybucji i instalacji w systemie operacyjnym, jak i później, na żądanie samego systemu operacyjnego lub też innej działającej w nim aplikacji. Tylko systemy Microsoft Authenticode oraz Apple Codesign umożliwiają wykonanie tak zwanej integracji tych systemów z językami programowania, wykorzystywanymi w procesie wytwarzania oprogramowania dla obsługiwanych przez te systemy platform programistycznych. W praktyce oznacza to, że wyłącznie systemy Microsoft Authenticode oraz Apple Codesign umożliwiają wykonanie weryfikacji tożsamości kodu uruchomionej aplikacji na żądanie samej aplikacji lub też na żądanie innej aplikacji działającej w danym momencie w systemie operacyjnym.

Bibliografia

1. Adams C., Lloyd S.: Understanding PKI: Concepts, Standards, and Deployment Considerations. Addison-Wesley Professional, New York 2002.
2. Apple Mac OS X Developer Library: Code Signig Guide. http://developer.apple.com/library/mac/ [dostęp: 23.03.2011].
3. Bond M., Robinson E.: Security for Microsoft Visual Basic .NET. Microsoft Press, Redmond 2003.
4. Brink D., Duane B., Joseph C., Nash A.: PKI: Implementing & Managing E-Security. McGraw-Hill Osborne Media, New York 2001.
5. Dessart F., Karamanian A., Tenneti S.: PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks. Cisco Press, Indianapolis 2011.
6. Garfinkel S., Spafford G.: Web Security, Privacy and Commerce. O'Reilly Media, Sebastopol 2002.
7. Gong L., Schemers R.: Signing, Sealing, and Guarding Java Objects. Springer Berlin-Heidelberg, Berlin 1998.
8. Grimes R., Microsoft TechNet Archive: Authenticode, http://technet.microsoft.comlivepage.apple.com/en-us/library/cc750035.aspx [dostęp: 20.03.2011].
9. Schneier B.: Kryptografia dla praktyków: Protokoły, algorytmy i programy źródłowe w języku C. Wydawnictwa Naukowo-Techniczne, Warszawa 2002.

10. The Java Tutorials: Signing Code and Granting It Permissions, http://download.oracle.com/javase/tutorial/security/toolsign/index.html [dostęp: 17.03.2011].

Abstract

In many business sectors software security is essential. One of the methods allowing to ensure consistency of software and required level of security is taking advantage of code signing systems. Use of these systems allows both software developers and software users to secure the software during its distribution and operating.

This publication discusses leading solutions for code signing systems such as Microsoft Authenticode (dedicated for Microsoft .NET framework), Oracle Java Code Signing (dedicated for Oracle Java platform) and Apple Codesign (dedicated for Apple Cocoa framework). All of these systems were compared to each other in aspects such as system features, digital signature structure, its generation and verification. Different aspects of code signing systems like modular signatures, dynamic and static digital signature verification has also been described. A special attention was given to an issue of code identity and its impact on code signing systems.