Učinkovitost kriptografskih funkcij v spletnih aplikacijah

Jasmin Malkić (jasmin.malkic@uni-mb.si)
Tatjana Welzer (welzer@uni-mb.si)
Boštjan Brumen (bostjan.brumen@uni-mb.si)
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za informatiko

Povzetek

Skladno z rastjo interneta se povečuje ponudba kriptografskih procedur. Kljub temu so vse procedure in aplikacije zasnovane na določenem številu kriptograpfskih konceptov in operacij.
Članek temelji na ocenjevanju časovne učinkovitosti nekaterih osnovnih kriptografskih operacij pri implementaciji v programskem jeziku Java. Merjenje učinkovitosti vključuje analizo dolžine ključa in generiranje digitalnega podpisa v simuliranju internetnega okolja. Pridobljene rezultate lahko uporabimo za izboljšanje časovne učinkovitosti spletnih aplikacij za varnen prenos podatkov.

Abstract

Performance of Cryptographic Functions in the Internet Applications

The choice of cryptographic procedures applicable on the Internet increases together with the Internet itself. However, all these procedures and applications are based on a few cryptographic concepts and known operations.
The present paper deals with the evaluation of some basic cryptographic operations by means of measuring time performance of their implementation in the Java programming language. The measuring of performance includes analyses of key length and generating of digital signature in the simulation of Internet working environment. The values obtained can be used for improving time performance of Internet systems for secure data transfer.

1 Uvod

Tradicionalna kriptografija na podlagi simetričnega ključa omogoča hitro in zanesljivo enkripcijo. Ker uporablja isti ključ pri enkripciji in dekripciji, se pojavlja problem varne izmenjave ključa. Problem je še posebno očiten, ko govorimo o javnih porazdeljenih računalniških sistemih, kot je internet, kjer se izmenjava izvaja po nezavarovanem komunikacijskem kanalu. Glede na to, da je tajnost ključa bistvena pri zagotavljanju zasebnosti takšne komunikacije, pravimo takšnemu konceptu kriptografija tajnega ključa [5].

V zgodnjih 70. letih sta Diffie in Hellman razvila drugačen koncept s parom ključev za enkripcijo in dekripcijo. Tisto, kar se je začelo kot algoritem za varno izmenjavo tajnega ključa (ki se še vedno pogosto uporablja), je na koncu dalo matematično osnovo za popolnoma nov kriptografski koncept, v katerem se digitalni zapis, kodiran z enim, dekodira le z drugim ključem iz istega para ključev.

V tem primeru se lahko ključ objavi v javnosti in se imenuje javni ključ. Sporočilo, kodirano z uporabo javnega ključa, je lahko dekodirano le z drugim ključem iz para, ki se imenuje privatni ključ, in ga hrani lastnik. Ena od matematičnih osnov za kriptografijo javnega ključa je problem faktorizacije velikega celega števila za iskanje vseh praštevil, ki ga enakomerno delijo. To ni edini primer uporabe kriptografije javnega ključa. Včasih ne potrebujemo enkripcije sporočila, temveč zagotovitev njegovega izvora. Ta problem rešuje koncept kriptografije javnega ključa, imenovan digitalni podpis (angl. digital signature). Če pri podpisovanju sporočila uporabimo tajni ključ, ga lahko preverimo le s primernim javnim ključem. Tako kodirano besedilo je digitalni podpis sporočila in je navadno pri prenosu združeno z izvirnim sporočilom, tako da lahko naslovnik preveri podpis s primerjavo izvirnega sporočila z dekodiranim podpisom pošiljatelja. To je zgolj osnovna zamisel. Dejansko je treba namesto podpisovanja celotnega sporočila podpisati samo njegov MD5 (angl. Message Digest 5) oz. SHA-1 (angl. Secure Hash Algorithm 1) izvleček. Algoritmi za digitalni podpis in njihove procedure generiranja para ključev so podrobno opisani v uradnem poročilu "Digital Signature Standard- FIPS PUB 186-2".

Najbolj pogost algoritem za to vrsto enkripcije je DSA (angl. Digital Signature Algorithm), ki je skupaj

z RSA (angl. Rivest-Shamir-Adleman) in ECDSA (angl. Rivest-Shamir-Adleman) priporočen kot državni standard za digitalne podpise ZDA [8].

Če upoštevamo vse navedeno, ni težko predvideti, da bodo dodatni kriptografski postopki še upočasnili prenos podatkov po omrežju. Pri snovanju varnega internetnega sistema, ki uporablja kriptografijo, je bistvenega pomena izbor prave kriptografske metode in orodij, ki bodo časovno učinkovita. Operacije kriptografije ključa, ki jih mora internetni varnostni sistem opraviti v ustreznem času (in jih je treba analizirati), so:

- generiranje tajnega ključa,
- simetrična ekripcija velikih datotek in sporočil,
- asimetrična enkripcija tajnega ključa,
- digitalno podpisovanje sporočila ali njegovega izvlečka.

Algoritmi za kriptografijo javnega ključa veljajo za bolj počasne v primerjavi z algoritmi, ki uporabljajo tajni ključ. Zato je treba analizirati dolžino kriptografskih ključev za kriptografijo javnega ključa in generiranje digitalnega podpisa. Prva meritev za izboljšanje učinkovitosti kriptografije javnega ključa je večinoma zagotovitev enkripcije manjše količine podatkov. Istočasno je treba zagotoviti, da varnost celotnega sistema ni ogrožena. Prav to je doseženo s podpisovanjem izvlečka sporočila (npr. 20 bytov pri SHA algoritmu) celotnega besedila. Krajši javni ključi pomenijo hitrejšo enkripcijo, ki je primerna za zelo obremenjene pošiljatelje vendar pa predstavlja tveganje za varnost sistema.

Takšne analize se lahko pridobijo le na podlagi točno določene programske implementacije kriptografskega protokola. Sam program je implementiran v objektno orientiranem programskem jeziku Java, ki ima nujno potrebno infrastrukturo ne le za kriptografijo znotraj svojih mrežnih aplikacij, temveč tudi za merjenje učinkovitosti v različnih računalniških okoljih.

2 Standardna in alternativna kriptografija v Javi

Pred pojavom javanskega standardnega razvojnega okolja (JSDK) 1.4.0 so bili javanski kriptografski razredi razdeljeni na "standardne" in "razširitvene". Standardni razredi, imenovani kot dobavitelj storitev kriptografije SUN (angl. cryptography provider), so vključevali podporo za izvleček sporočila, MAC (angl. Message Authentication Code), digitalne podpise DSA in certifikate formata X.509.

Preostanek kriptografskih razredov je bil zapakiran v javansko kriptografsko razširitev (angl. Java Cryptography Extension, JCE) SUN, ki je vsebovala šifrirno orodje za kriptografijo tajnega ključa (angl. symmetric cipher), protokol za izmenjavo tajnih ključev in nekatere posebne implementacije za MAC. Obstajalo je nekaj razlogov za delitev kriptografskih funkcij, med katerimi so posebne izvozne omejitve v ZDA in pogoji za uporabo kriptografije javnega in tajnega ključa (zato JSDK ni bil zmožen enkipcije sporočila).

JCE je pokrival funkcije kriptografije tajnega ključa, posebna implementacija orodja za kriptografijo javnega ključa je bila vključena v drugo javansko kriptografsko razširitev SSL (angl. Java SSL Extension, JSSE), ki je bila namenjena za razvijanje sistema, ki implementira SSL (angl. Secure Socket Layer). Problem z uporabo množičnih razširitev skupaj s standardnimi paketi je bil potreba za dodatno integracijo razširitvenih paketov v sestavljena javanska strežniška okolja (npr. javanski aplikacijski strežniki za servlete).

JSDK 1.4.0, ki je bil dokončan februarja 2002, je prinesel popolno kriptografsko funkcionalnost znotraj svojih standardnih paketov. To dejstvo lahko prihrani čas za konfiguracijo razširitev in ponudi učinkovitejšo kriptografijo z uporabo dodatnih datotek, ki vsebujejo posebno politiko pristojnosti ter dovoljuje dolžine ključev AES (angl. Advanced Encryption Standard) in RSA 128 in 2048 bitov (če to dovoljujejo lokalne oblasti).

Implementacija alternativnega dobavitelja javanske kriptografije praviloma pomeni prepisovanje originalnih SUN javanskih razredov. Zaradi tega so alternativni JCE paketi navadno pisani za posebne namene, ko aplikacija potrebuje algoritem, ki ga SUN ni implementiral ali ga je implementiral na način, ki ne ustreza potrebam.

Da bi bil alternativni JCE operativen, morajo biti arhivske datoteke, ki vsebujejo njegove razrede, vpisane v lokalni sistemski spremenljivki „classpath". Naslednji korak je postavljanje dovoljenj in vpis novega dobavitelja kriptografije v lokalnih javanskih varnostnih mehanizmih in pristojnih datotekah. Da bi delovali v okviru varnostnega sistema, morajo biti novi razredi vpisani poleg ustreznih dovoljenj. Sam dobavitelj kriptografije je lahko vpisan statično v varnostni datoteki (ki lahko vsebuje nekaj dobaviteljev kriptografije) ali dinamično v programu, ki ga uporablja. Klicanje metode za vpis dobavitelja iz programa zahteva manj administrativnega dela, vendar je ta vpis veljaven samo v tem programu.

Nekaj organizacij in podjetij ponuja implementacije alternativnih JCE za izobraževalne namene ali

komercialno uporabo, npr. IBM [3], RSA Laboratories [11], IAIK (TU, Graz) [9], Bouncy Castle Group [10] itn. Te imajo navadno poudarek na izboljšavi izbire kriptografskih algoritmov za objekte tipa Cipher ali Key, kar prizadene te razrede v izvirni implementaciji. Podjetje SUN pomaga pri vključevanju abstraktnih razredov vmesnika za dobavitelje servisov (angl. Service Provider Interface, SPI) za glavne kriptografske objekte, kot so npr. CipherSpi, KeyGeneratorSpi itn. v javansko standardno razvojno okolje. Tisti, ki želijo razviti alternativne dobavitelje kriptografije, lahko implementirajo vse abstraktne razrede in metode, ki so v SPI.

3 Aplikacija za merjenje učinkovitosti javanskih kriptografskih funkcij

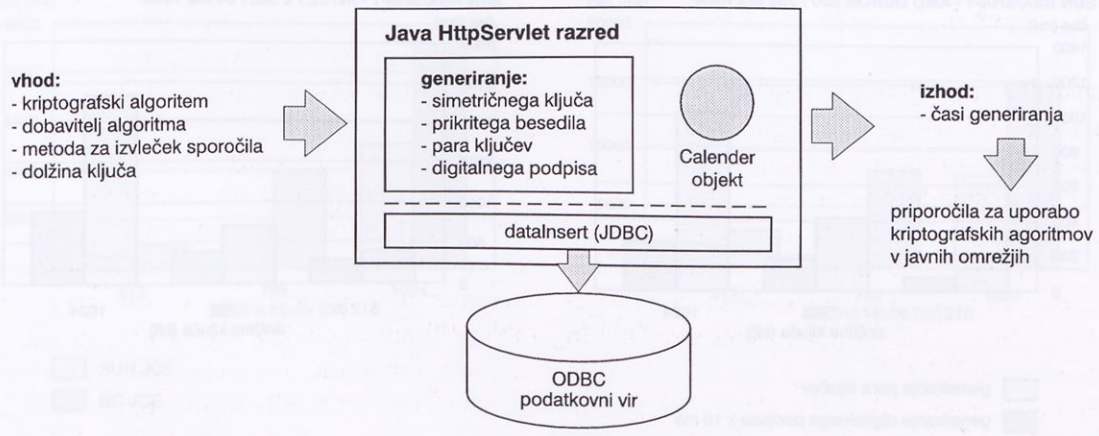
Da bi izmerili učinkovitost izvajanja programskih funkcij na določenem operacijskem sistemu, moramo pravilno uporabiti sistemsko uro. Podpora za časovne objekte (manipulirajo sistemsko uro) v programskom jeziku Java je podana v nekaj razredih, kot sta npr. Date in Calendar. Ker zaradi zastarelosti ne priporočamo uporabe razreda Datum, uporabljamo pri tem testu object tipa Calendar, ki zabeleži lokalni čas točno ob svojemu nastanku. Aplikacija za merjenje učinkovitosti Java kriptografskih funkcij kreira skupno tri primerke objekta Calendar, med katerimi se izvedejo kriptografske operacije, ki jih merimo.

V tem primeru prva vrednost zabeleži nastanek para ključev za algoritem javne kriptografije. Kot vstopne parametre aplikacija vzame algoritem in dobavitelja, za katera se ustvarijo ključi. Druga časovna vrednost vsebuje čas za vpeljavo objekta tipa Signature in proces digitalnega podpisovanja (metoda za izvleček sporočila, ki ga podpisujemo, je vključena v vhodne parametre). Pomemben vhodni parameter za aplikacijo je tudi dolžina ključa. Analiza za algoritem javnega ključa je bila izvedena za dolžine 512, 768 in 1024 bitov. Ta razprava lahko da pregled javanske kriptografije, ki je trenutno v uporabi. Kriptografski ključi dolžine 2048 bitov so dovoljeni samo v najnovejših različicah javanskih delovnih okolij pod posebnimi pogoji (ta uporaba vključuje posebne datoteke za konfiguracijo javanske kriptografske razširitve).

Testna aplikacija razširja HttpServlet razred, ki ga kličemo z http zahtevo (ta vsebuje vhodne parametre) na lokalnem aplikacijskem strežniku za javanske servlete. Njen cilj je meritev učinkovitost v realnemu času in pogojih dinamičnega spletnega okolja. Izid javanskega servleta (slika 1) v http obliki je usmerjen v spletni brskalnik, iz katerega je prišla http zahteva. Rezultati merjenja učinkovitosti se beležijo v lokalni bazi za nadaljnjo analizo. Operacija je izvedena s klicem metode dataInsert, ki lahko izvede vsako poizvedbo na lokalnem ODBC (angl. Open Database Connection) viru podatkov. Izvirna koda metode in celotne testne aplikacije se nahaja na spletnem naslovu http://www.inet.ba/malkic/ird2.

4 Učinkovitost in varnostna analiza

Načrtovanje varnostnega sistema, ki uporablja javansko kriptografijo, zahteva natančno poznavanje kriptografskih orodij, ki naj bi se uporabljala. Za pravilno izbiro kriptografskih algoritmov in dolžino ključev so zaželeni tudi natančni podatki o zmogljivosti posameznih implementacij kriptografskih orodij. Da bi



Slika 1: **Vhodi in izidi testne aplikacije**

dobili takšne podatke, smo izvedli test z zgoraj opisano javansko spletno aplikacijo.

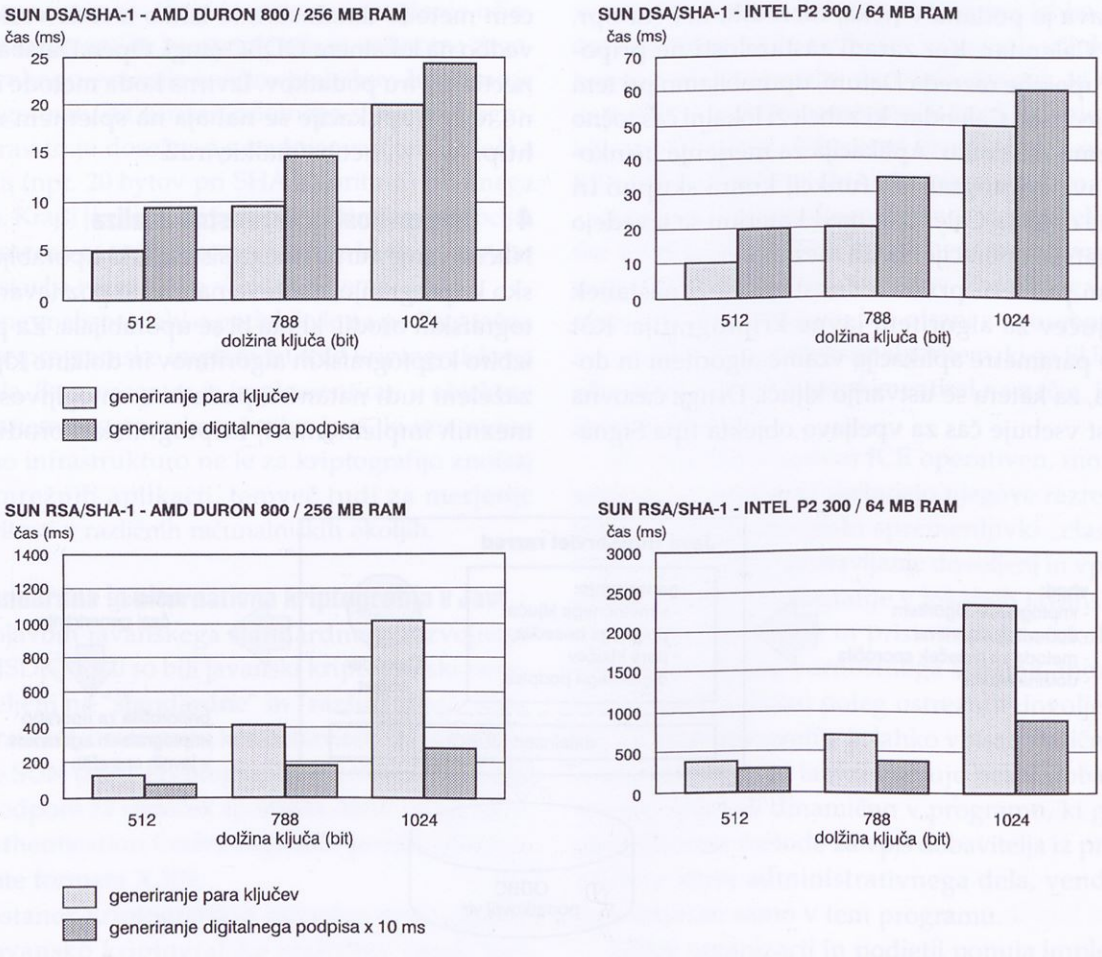
Različni algoritmi za digitalni podpis so podvrženi testiranju/merjenju najpomembnejših karakteristik:

- časa, potrebnega za generiranje para ključev za algoritem kriptografije javnega ključa in
- časa, potrebnega za proces izdelave digitalnega podpisa tekstovne datoteke skupne dolžine 12.650 bajtov na lokalnem trdem disku. V tem delu eksperimenta sta vključena DSA in RSA algoritma iz SUN javanske kriptografske razširitve z uporabo SHA-1 izvlečka sporočila za digitalni podpis.

V primeru enostavnega avtentikacijskega namena ni mogoče pričakovati generiranja para ključev (sistem z več kompleksnosti lahko potrebuje takšno funkcijo). Čas, potreben za operacijo digitalnega podpisovanja, je lahko zelo pomemben pri načrtovanju spletnega kriptografskega sisitema v realnem času. Merjenje je torej narejeno za tri dobavitelje javanskih kriptografskih razširitev – standardni SUN in dve alternativni, IAIK in Bouncy Castle. Omeniti je treba, da je za SUN in Bouncy Castle implementacijo Diffie-Hellmanovega algoritma za ujemanje ključev merjen samo čas generiranja para ključev.

Žal je uporaba nekaterih razširitev tega tipa omejena z njihovimi komercialnimi licencami. IAIK, ki ga je razvil "Institute for Applied Information Processing and Communications, Graz University of Technology", se lahko uporablja za potrebe izobraževanja [9], medtem ko je Boncy Castle kriptografski paket v prosti uporabi [10].

Testiranje na aplikacijskemu strežniku Apache Tomcat 4.1 z dvema kombinacijama strojne opreme naj bi pokazalo, da je relacija med dolžino kriptografskega ključa in časom, potrebnim za operacijo digitalnega podpisovanja z uporabo tega ključa, neodvisna od izbire strojne opreme. Iz slik 2 in 3, je razvidno, da počasnejša strojna oprema potrebuje bistveno več

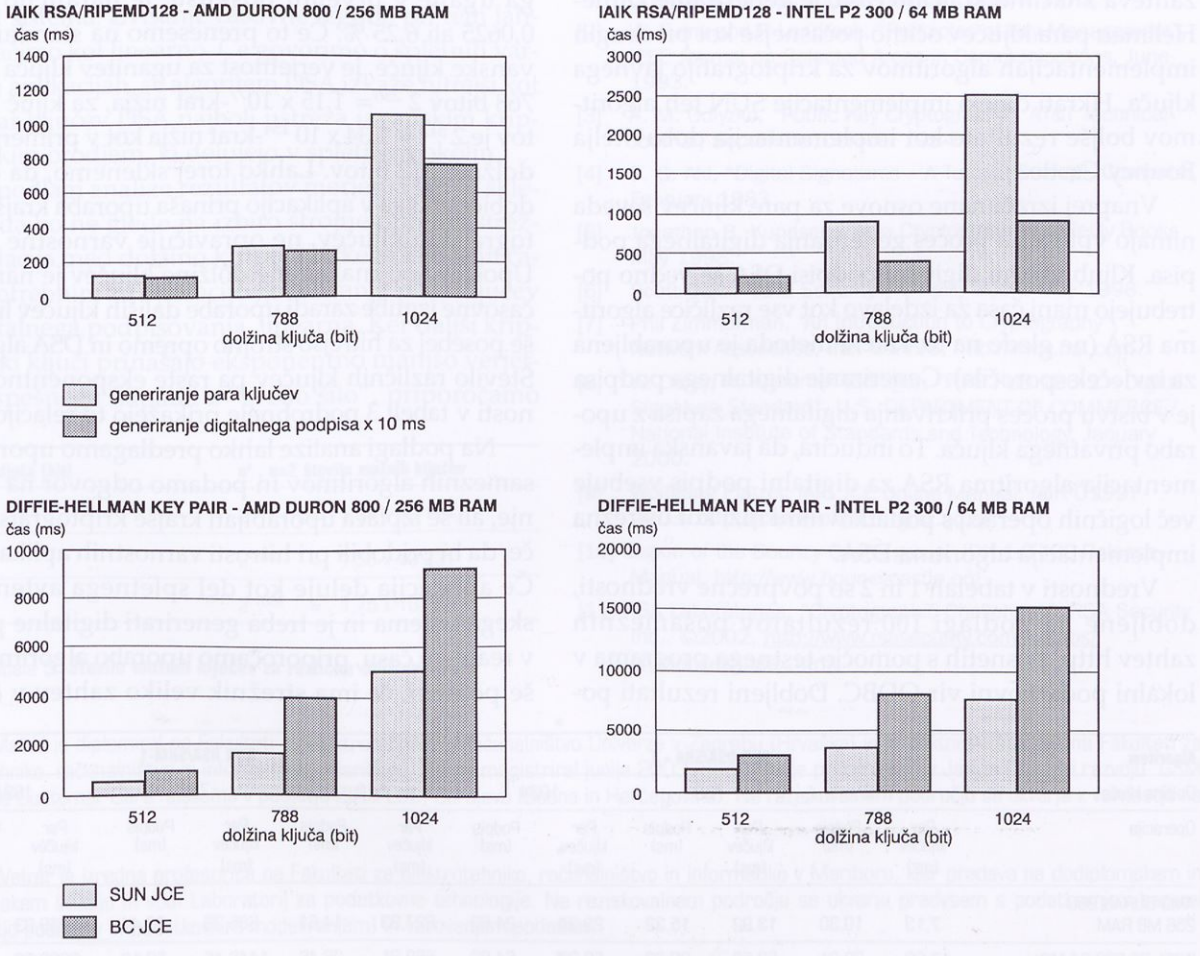


Slika 2: **Vrednosti časa za kriptografske operacije SUN algoritmov za kriptografijo javnega ključa**

časa za izvajanje kriptografskih operacij. Celo iz takšne grobe predstavitve rezultatov lahko sklepamo, da strojna oprema nima značilnega vpliva na trend povečanja časa, ki ga potrebuje proces generiranja para ključev, s povečevanjem dolžine ključa. Izvajanje procesa digitalnega podpisovanja je lahko namreč bolj odvisno od izbire strojne opreme, ker mora testni program v tem primeru napolniti objekt tipa Signature do dolžine 12.650 bajtov podatkov s trdega diska. Program izvede ta proces v 12 delih dolžine 1024 bitov in enemu delu dolžine 362 bitov. Spremenljivi čas dostopa do trdega diska lahko povzroči dodatne časovne izgube, vendar nam merjenje v takšnih pogojih lahko da pregled lastnosti procesa digitalnega podpisovanja. Natančni rezultati merjenja v tabelah 1 in 2 predstavljajo podlago za končno sklepanje.

Izbira algoritma za izvleček sporočila vpliva tudi na izdelavo digitalnega podpisa, saj ima krajši izvleček sporočila (128 bit RIPEMD, angl. RIPE Message Digest) boljši učinek od daljših (kot npr. 20 byte SHA-1). Da bi ta razlika vplivala na rezultate, je treba opazovati samo digitalni podpis izvlečka sporočila. V tem primeru mora biti operacija pridobivanja izvlečka sporočila narejena zunaj programskega bloka, na katerega vpliva merjenje.

Testna aplikacija z uporabo RSA para ključev v Java kriptografskih aplikacijah potrebuje za generiranje para ključev stokrat več časa kot uporaba DSA algoritma. Po drugi strani pa proces generiranja digitalnega podpisa povprečno vzame "samo" dvakrat več časa (zaradi lažje primerjave, so vrednosti za generiranje RSA digitalnega podpisa pomnožene z 10). Do tako velike razlike prihaja zaradi specifične javanske implementacije kriptografskih algoritmov. Za SUN dobavitelja kriptografije in DSA par ključev 512, 768 in 1024 bitov, ima razred KeyPairGenerator na voljo



Slika 3: **Čas, potreben za generiranje para ključev in digitalnega podpisa, za IAIK RSA in D-H algoritme ujemanja ključev**

že izračunan velik sklop primarnih celoštevilčnih vrednosti (ki so osnova za izračun parov kriptografskih ključev). Z uporabo sistemskega naključja in omenjene osnove, KeyPairGenerator lahko dela veliko hitreje kot v primeru drugih algoritmov, kjer se mora material za par ključev generirati v realnem času.

Načrtovalci alternativnega dobavitelja kriptografije lahko integrirajo vnaprej izračunano osnovo v svoje javanske kriptografske razširitvene pakete (JCE) s ciljem, da omogočijo hitrejše izvajanje tudi nekaterim drugim kriptografskim algoritmom. V primeru da zunanji faktorji (kot so zakonske omejitve) preprečujejo uporabo določene implementacije kriptografskega algoritma, se lahko programerske skupine, ki razvijajo varnostne aplikacije, odločijo tudi za razvoj lastnega kriptografskega razširitvenega paketa. Osnova za par ključev algoritma Diffie-Hellman mora tudi biti izračunana v realnemu času iz sistemskega generatorja naključnih števil in dveh velikih praštevil. Ker izdelava te osnove zahteva značilno količino časa, je generiranje Diffie-Hellman para ključev očitno počasnejše kot pri drugih implementacijah algoritmov za kriptografijo javnega ključa. Hkrati dajejo implementacije SUN teh algoritmov boljše rezultate kot implementacija dobavitelja Bouncy Castle.

Vnaprej izračunane osnove za pare ključev seveda nimajo vpliva na proces generiranja digitalnega podpisa. Kljub vsemu digitalni podpisi DSA še vedno potrebujejo manj časa za izdelavo kot vse različice algoritma RSA (ne glede na to, katera metoda je uporabljena za izvleček sporočila). Generiranje digitalnega podpisa je v bistvu proces prikrivanja digitalnega zapisa z uporabo privatnega ključa. To inducira, da javanska implementacija algoritma RSA za digitalni podpis vsebuje več logičnih operacij s podatkovnimi nizi kot ustrezna implementacija algoritma DSA.

Vrednosti v tabelah 1 in 2 so povprečne vrednosti, dobljene na podlagi 100 rezultatov posameznih zahtev http, posnetih s pomočjo testnega programa v lokalni podatkovni vir ODBC. Dobljeni rezultati potrjujejo ugotovitev, da sta celo z različno strojno opremo, časa generiranja para ključev in digitalnega podpisa odvisna od dolžine uporabljenega kriptografskega ključa. Vpliv strojne opreme je omejen na situacije, ko strojna oprema povzroči naključno prekinitev v pretoku podatkov (npr. v primeru generiranja digitalnega podpisa zaradi počasnega dotoka podatkov s trdega diska ali primanjkljaj procesnega časa, ki ga je povzročil "context switch"). Temu se lahko izognemo s hitrejšo in močnejšo strojno opremo ali s ponovljenim merjenjem na slabši opremi.

Glede na varnostne zahteve je treba omeniti, da ima vsak bit v ključu dve možni vrednosti; ključ ima x bitov, zato je število možnih različnih ključev 2^x. Večje število možnih ključev pomeni manjšo verjetnost za uganitev ključa, kar je osnovna operacija napada z grobo silo, pri kateri skušamo preveriti vsak možni ključ. Če je ključ npr. dolg 4 bite, bi ga tak program uganil po vsaj $2^4 = 16$ poskusih. Verjetnost, da ga ugane v prvemu poskusu, je natančno 1/16 = 0,0625 ali 6,25 %. Če to prenesemo na standardne javanske ključe, je verjetnost za uganitev ključa dolžine 768 bitov $2^{256} = 1{,}15 \times 10^{77}$-krat nižja, za ključ 1024 bitov je $2^{512} = 1{,}34 \times 10^{154}$-krat nižja kot v primeru ključa dolžine 512 bitov. Lahko torej sklenemo, da časovni dobiček, ki ga v aplikacijo prinaša uporaba krajših kriptografskih ključev, ne opravičuje varnostne izgube. Upoštevajoč analizirane dolžine ključev je naraščanje časovne izgube zaradi uporabe daljših ključev linearno, še posebej za hitrejšo strojno opremo in DSA algoritem. Število različnih ključev pa raste eksponentno. Vrednosti v tabeli 3 podrobneje prikažejo to relacijo.

Na podlagi analize lahko predlagamo uporabo posameznih algoritmov in podamo odgovor na vprašanje, ali se izplača uporabljati krajše kriptografske ključe, da bi pridobili pri hitrosti varnostnih aplikacij Java. Če aplikacija deluje kot del spletnega avtentikacijskega sistema in je treba generirati digitalne podpise v realnem času, priporočamo uporabo algoritma DSA, še posebej, če ima strežnik veliko zahtev v časovni

Algoritem	**SUN DSA/SHA - 1**						**SUN RSA/SHM-1**					
Dolžina ključa	512		768		1024		512		768		1024	
Operacija	Par ključev (ms)	Podpis (ms)	Par ključev (ms)	Podpis (ms)	Par ključev (ms)	Podpis (ms)	Par ključev (ms)	Podpis (ms)	Par ključev (ms)	Podpis (ms)	Par ključev (ms)	Podpis (ms)
AMD DURAN 900 256 MB RAM	7,13	10,30	13,92	15,32	23,34	24,83	227,83	14,61	685,38	23,44	1258,03	36,63
INTEL P2 300 64 MB	17,63	23,21	33,53	39,89	59,36	64,02	569,81	26,46	1446,45	50,19	2809,69	102,68

Tabela 1: **Povprečna vrednost časa, potrebnega za generiranje para ključev in digitalnega podpisa za SUN algoritme**

Algoritem	IAIK RSA/RIPEMD 128						Diffie-Hellman KA					
Dolžina ključa	512		768		1024		512		768		1024	
Operacija/JCE dobavitelj (ms)	Par ključev	Podpis	Par ključev	Podpis	Par ključev	Podpis	Sun	Bouncy Castle	Sun	Bouncy Castle	Sun	Bouncy Castle
AMD DURON 900 256 MB RAM	232,75	16,20	579,99	34,60	1238,79	62,55	862,18	1145,03	3665,36	4533,28	6795,70	9669,20
INTEL P2 300 64 MB RAM	576,25	23,25	1480,99	51,12	2865,05	104,02	2020,26	3160,25	6204,66	9281,61	11700,57	16526,55

Tabela 2: **Povprečna vrednost časa, potrebnega za generiranje para ključev in digitalnega podpisa za IAIK RSA in D-H algoritme ujemanja ključev**

enoti. Digitalni podpisi DSA potrebujejo manj časa za generiranje ne glede na strojno opremo. Seveda je v primeru manjših količin podatkov (kot so 128-bitni tajni ključi), priporočljiva tudi uporaba algoritma RSA.

5 Sklep

Ker se moč procesiranja sodobnih računalnikov hitro povečuje, je smiselno zvišati varnost sistema z uporabo najdaljšega možnega ključa. Tako se verjetnost za njegovo uganitev eksponentno zmanjšuje, kar zvišuje varnost sistema. Zvišanje časovne izgube pri tem lahko označimo kot linearno. Če govorimo o spletnih varnostnih aplikacijah, se algoritem DSA izkaže hitrejši kot RSA. Par ključev DSA najbolj ustreza javanskim kriptografskim orodjem, ki delujejo v spletnem okolju.

Na podlagi analize rezultatov merjenja lahko sklepamo, da bo ne glede na izbiro strojne opreme strežnika relacija med dolžino kritpografskega ključa in časom, potrebnim za operacijo generiranja para ključev ali digitalnega podpisovanja, linearna. Ker daljši kriptografski ključi prinašajo eksponentno manjšo verjetnost uspešnega napada z "grobo silo", priporočamo njihovo uporabo. Če govorimo o digitalnih podpisih v programskemu jeziku Java, je zaželena uporaba kriptografskega algoritma DSA. Izkazalo se je tudi, da potrebuje proces generiranja para ključev za algoritem Diffie-Hellman ujemanja ključev bistveno več časa, kot druge javanske implementacije algoritmov za kriptografijo javnega ključa.

x, dolžina ključa (bit)	n^x, n=2 število možnih ključev
512	2^{512} = 1,34 x 10 124
768	2^{768} = 1,55 x 10 231
1024	2^{1024} = 1,75 x 10 308

Tabela 3: **Število možnih ključev za različne dolžine ključa**

Viri

[1] M. E. Hellman, "The Mathematics of Public-Key Cryptography", Scientific American, August 1979.

[2] W. Fumy and P. Landrock, "Principles of Key Management", IEEE Journal on Selected Areas in Communications, June 1993.

[3] A. M. Odlyzko, "Public Key Cryptography", AT&T Technical Journal, September/October 1994.

[4] S. G. Akl, "Digital Signatures – A Tutorial Survey", Computer, February 1983.

[5] Jonathan B. Kundsen, "Java Cryptography", O'Reilly Books May 1998.

[6] Scott Oaks, "Java Security", O'Reilly Books May 1998.

[7] Phil Zimmerman, "An Introduction to Cryptography", Network Associates, Inc. ©1998, http://www.nai.com

[8] W. Daley, R. Krammer (Editors), "FIPS PUB 186-2 – Digital Signature Standard", U.S. DEPARTMENT OF COMMERCE / National Institute of Standards and Technology, January 2000.

[9] Wolfgang Platzer, IAIK JCE Online Manual, IAIK Ó1997-2003, http://jcewww.iaik.tu-graz.ac.at

[10] Legion of the Bouncy Castle group, JDK 1.3 JCE Release Manual, http://www.bouncycastle.org

[11] RSA Laboratories, "Cryptographic Challenges", RSA Security Inc., ©2002, http://www.rsasecurity.com/rsalabs/challenges/index.html

Jasmin Malkić je diplomiral na Fakulteti za elektrotehniko in računalništvo Univerze v Zagrebu (Hrvaška) in je doktorski študent na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru, kjer je magistriral junija 2003. Sodeloval je pri projektih v Javi in C++ pri razvoju "GSM Billing and Customer Care" sistema v podjetju ZIRA Ltd., Sarajevo (Bosna in Hercegovina). Na raziskovalnem področju se ukvarja z varnostjo na internetu.

Tatjana Welzer je izredna profesorica na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru, kjer predava na dodiplomskem in podiplomskem študiju in vodi Laboratorij za podatkovne tehnologije. Na raziskovalnem področju se ukvarja predvsem s podatkovnimi bazami (kakovostjo podatkov in podatkovnim modeliranjem) in varovanjem podatkov.

Boštjan Brumen je doktorski študent na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru, kjer je zaposlen kot asistent za področje informatike. Na raziskovalnem področju se ukvarja s podatkovnimi bazami, podatkovnim rudarjenjem in varovanjem podatkov.