Трета домашна задача по информациска безбедност

Марија Трпчевска

196002

Документација за имплементацијата на STS во пракса

1. Иницијални подготовки

Во рамки на проектот се дефинирани пет класи: Main, AES, RSA, User и CertificateAuthority. Во Main се одвива конкретна инстанца на STS протоколот, AES во CBC мод со padding служи при симетричното шифрирање со разменетиот клуч додека со RSA се креираат асиметричен пар клучеви користени при потпишувањето. Во User класата се опишани сите акции кои корисниците ги преземаат во протоколот додека CertificateAuthority служи за издавање и верифицирање на сертификати во обликот потребен за STS (бидејќи вклучува и дополнителни информации во сертификатот покрај идентификација на корисникот и неговиот јавен клуч, потребна е посебна дефиниција со што користењето на постоечката Java класа Certificate е невозможна (поддржува стандарди како што е X509, но не и кориснички дефинирани ‘стандарди’).

1. Иницирање на протокол Alice → Bob

Во Main се инстанцираат две инстанци од User, Alice и Bob, a атрибутот RSAkeyPair е пополнет со повикување на методот generate\_key\_pairs() од RSA, при што се креира приватен и јавен клуч, двата долги 2048 бита соодветно. Истиот метод е повикан и за CertificateAuthority кој има статички атрибути и методи бидејќи се претпоставува дека истиот CA ги има сертификатите за Alice и Bob. Преку Alice се повикува методот initiate\_protocol() кој за циклична група секогаш ја зема вредноста 7, а за генератор вредноста 5 бидејќи (50(mod7),51(mod7),52(mod7),53(mod7),54(mod7), 55(mod7)) = (1,5,4,6,2,3), па 5 е генератор (g) на {1,2,3,4,5,6} односно циклична група (p) од ред 7 (имање на посебен метод кој за случајна циклична група ќе бара генератор значително ќе ја зголеми пресметувачката комплексност на проектот). Експонентот за Alice (a) се бара како случаен број меѓу 10 и 100 (овој ранг произведува големи вредности за клуч понатаму, но не до толку големи за да се ризикува излегување од рангот на вредности за BigInteger и настане overflow). Се пресметува ga и трите вредности се пакуваат во листа и праќаат до Bob. Да се напомене дека се работи со BigInteger тип за сите нумерички вредности во рамки на проектот и во атрибути покрај RSAkeyPair се чува DHKey кој е тајниот клуч, својот експонент, вредноста на генераторот, цикличната група и пратениот gx од соговорникот.

1. Прифаќање на иницираниот протокол Bob → Alice

Bob преку parse\_request() ги добива пратениот генератор и циклична група, на ист начин пресметува свој експонент (b), и го пресметува клучот DHKey = (ga)b. Потоа повикува инстанца на Signature класата од Java.security во SHA512/256withRSA мод (STS бара потписот да биде дополнително и хеширан), го иницијализира потписот со јавниот клуч од RSAkeyPair парот и пресметаниотgb со добиениот ga во тој редослед ги става во ByteBuffer (бидејќи потпис може да се врши само врз бајти, низа од бајти или бафер со бајти), ги потпишува бајтите во баферот и истите потоа ги шифрира користејќи го DHKey со AES.

Следи генерирање на сертификат преку методот generateCertificate()од CertificateAuthority класата кој го прима идентификаторот за корисникот (String ‘Bob’ конкретно), генераторот, цикличната група и јавниот клуч од RSAkeyPair на Bob), иницијализира Signature инстанца со својот приватен клуч и ги става сите вредности во бафер, ги потпишува и со добиениот потпис заедно со претходните вредности инстанцира STSCertificate објект кој го праќа како одговор. Конечно, во Object низа се ставаат gb, сертификатот и шифрираните gb, ga и се праќаат до Alice.

1. Верификација Alice → Bob

Преку Alice се повикува verify\_communication() која ја прифаќа низата од објекти пратени од Bob и го праќа добиениот сертификат преку verifyCertificate() до CertificateAuthority каде се инстанцира Signature класата и иницијализира, но овојпат со јавниот клуч на CA, се прави бафер од вредностите (освен потписот) и истиот се верификува за еднаквост со потписот. Ако се врати true, се продолжува со пресметка на DHKey = (gb)a кој потоа се користи за дешифрирање со AES на шифрираните податоци. Следи потврда на потписот на Bob за која цел инстанцираниот потпис се иницијализира со добиениот јавен клуч од сертификатот, се креира бафер со добиената вредност за gb и својата вредност ga па се проверува за еднаквост со потписот. Ако се врати true, Alice со својот приватен клуч ги потпишува gа, gb во тој редослед, побарува од CertificateAuthority издавање на сертификат и двете вредности како низа објекти ги праќа до Bob.

1. Верификација Bob → Alice

Bob со повикување на verify\_communication() го верификува сертификатот на Alice, креира потпис со добиениот јавен клуч, баферира и го верификува потписот на Alice. Ако сите проверки не јават проблем успешно се завршува STS протоколот.

\*Да се напомене дека може verify\_communication() да се изведе и на страната на User со јавниот клуч на CA, но тоа е изведено на страна на CA во решението за помодуларен пристап/прегледност. Јасно, во пракса ова ќе го доведе CA во улога на тесно грло бидејќи е принуден да учествува во секоја инстанца на протоколот.

1. Пример излез од програмата

GENERATOR: 5

CYCLIC\_GROUP: 7

ALICE EXPONENT A: 60  
ALICE G^A: 867361737988403547205962240695953369140625

BOB EXPONENT B: 40  
BOB G^B: 9094947017729282379150390625

SHARED SECRET KEY:

3372953907415801999510547124727770258842113978259391260403834163166850051218632691553041346799279882981515297557541570964112773226923960801537836011613761666380973748253140846836385540541656893533566556532128587609119972043949691309449703242115013393419540459927102579498346467299401012749720009304462419499298338826121665091807352081431242810848969392739405430719210271501023463741153813750941579861423621799559857511832858485197337111670542015421403592476800302914134457106830800029293386001669530945721080546298578522004661341363026572512318855267198898327898377575859120532432417397069339119811619412262841079909982797622055202242346170402549106320605812844545336042293857785741660963970122720985979332480034177089701298715185851026982329340510387898360312460148941463348606289453998692820878591746083049646551653713327269442369616980120893977866832460484320084373729604736216817912509518838660159007080891829026557314178400690536703566917082048206015374630856389229787403236090417391289495435638473216641854902691125994990059820180806937920474197940940875134353774635162516263215724111373948579714996396603447040078923168989282910183705974082427671553597811123390443611343856048610631903083743675836811291453305994170156783637924121313526268479876366300942053350229051287416788009250739541127640572355538960827254770773149902720123489722932963818701863929421668573926878495945940491037858406926763386752542759682155019491287140371733444137555315644240768644578833449403697685562485213294430860562343818907827393883080527109656977165077992618536836166084869536934178823682425538422740781625687031197465531189508951823552257611102942781722280329859131597913801670074462890625

BOB SIGNED PAYLOAD: c 1d d1 b6 4e 1 ae 22 d7 7d e1 cb a2 42 7a 61 84 bc b9 45 dd aa da a2 a4 a3 c2 e7 2d 4 ef f4 d0 7c 6c 24 e7 c2 d1 65 d1 82 7c 7 8a bb 1a de 95 4d 19 37 21 c7 56 12 44 8e fa 27 fc 96 dd 56 5a 1d da 81 f2 b4 d9 14 43 8e 17 1c ce 46 97 42 91 a8 f8 53 7b 0 bf 88 1f 45 15 61 9a b7 a7 8c 6 81 1f a6 68 87 8d b2 30 9b 7 17 b5 da b7 c1 aa 34 3f c2 fc 3d 89 22 18 61 b7 75 49 35 2 c5 a8 82 2a 5 85 13 b6 8b 46 8b 99 6e b1 db 44 e 84 58 7a e9 1e 84 16 3c 63 60 1 85 c6 db 54 88 1d a c4 13 be 59 c4 9e 16 96 aa ae ee 7c d4 cc a0 d0 21 f4 8 3a d1 18 3d 1b f0 31 83 8 dd c2 7b 18 23 41 73 39 1f 15 69 d0 50 6f 75 1d 2e d3 b9 e7 f0 64 78 cd b2 7d 3a c6 7a 2e 73 33 d 93 a5 45 54 68 a5 9c 9c b8 2e 61 d2 d7 ab f2 40 f6 35 fb 83 f3 de 2c 58 8e 94 93 64 a1 5c 5 ae bf

BOB ENCRYPTED PAYLOAD: 18 3b 1b 76 7c bf d1 90 ba 6b c6 d0 47 be df b1 15 88 90 24 5c 50 5c 4b 7b f1 89 44 47 cb 64 49 c2 e1 fe 15 3 c9 28 e0 fb b6 a5 ae 90 cc cd d7 e c5 ba f3 1a 48 73 d2 b 99 f3 95 8f 2f 98 ff 1 1 b6 50 9b e4 54 25 b2 85 60 17 3a 80 41 10 77 9e bd c 28 93 f9 bb 83 78 13 90 ed 8a d5 c7 ad e6 31 b0 32 e0 a1 d4 d9 85 98 34 78 d5 eb ea ca c aa 84 27 22 85 ed ed 5b 1 f7 78 eb f6 10 b1 6e cf 11 16 f0 fe 44 3a 3a 3c ee 8b 48 49 8b 6f 74 cb fa 3f 8a 3a 7c 98 36 fc 9c b6 4b c0 af d7 f5 cb af 23 33 7 bc f5 27 87 2c 55 44 5c 5d 70 b3 c0 b9 f1 5a 18 7 b7 9 a6 a7 ce fc 36 d 4e f7 25 b0 19 b2 dc c1 8f 50 c6 72 e0 2 12 84 64 d3 32 33 ba 52 c7 77 63 e4 cf 78 80 5f af 28 f1 c9 8d 31 7d 69 94 46 92 dd 81 b5 8c 8b b4 d4 68 ea af 29 ca 4a d7 f9 96 32 51 2b 3e 33 35 fe df 45 ad a b5 b6 12 f7 52 a4 72 69 45 8a 16 1d

CA SIGNATURE OVER BOB'S INFORMATION (ID, G, P, PUBLIC\_KEY): 43 f2 e0 18 9d 7c 23 63 24 c1 2f 57 ad 16 da a8 b1 40 9a d3 fd 6f 85 82 7a f2 f4 90 60 c4 69 30 ab 41 e3 b d1 a9 82 e1 80 c1 ce de 15 a6 99 7f 85 6d 7c 7c 81 74 59 20 1d 1 81 f4 d2 cf cc 6e ed 8a 7 96 c4 fd a6 90 fd 3d 88 39 1e 11 37 91 cb ef 79 df 0 55 88 62 9a b4 44 33 87 1f 72 d4 f1 72 5c 28 2a 97 7f 66 b0 be 35 71 c2 fa 72 84 66 fc bd dc 65 c3 f0 d2 40 bb e1 76 96 91 51 2e 72 d5 7f ff d9 67 71 70 0 8e 38 8f e0 1f ee c6 a8 44 74 17 bc 9a 3c 35 fc 8e f3 e3 43 d6 e9 2a c7 5a 9e 3b 48 61 fe ba 91 f1 e5 1e 86 fe eb 81 e 5a 4d d1 24 61 8c 2 ae 3e ef 4a 4a 63 e6 36 85 fd 68 b9 16 78 75 73 d6 d a7 c1 4f f4 79 a3 6b c6 3c c7 d4 51 58 34 84 8f fc 7d 3e 22 cf 94 3d 7a c2 79 d0 c8 1a 85 9d 6c 2a 6d c0 51 97 32 e1 fc fc 85 af 1d 63 82 f2 45 c8 6c 2d 9 14 c0

VERIFIED CERTIFICATE SUCCESSFULLY FOR Bob

VERIFIED SIGNED PAYLOAD SUCCESSFULLY FROM Bob

ALICE SIGNED PAYLOAD: 2 97 91 7b de 48 2d b9 8f bb 36 35 c de b7 f0 d4 5f 3e f4 71 59 ee dc 8 4e 72 36 79 db 4e 63 1a 2c 90 7d 4f 76 3c cf fc 4a 86 6f 40 46 dd fb 79 6e aa 1b c4 17 d6 60 a7 9c 72 e1 fc 84 9e 52 94 79 3 3 d9 9b 4e fe de 6c d0 5f 56 13 f 72 75 bb 90 9 74 43 bb 2 3a 9f df e8 1e 71 1b 2d e4 75 b a5 5a 5b 56 80 57 3e c8 4 b7 cd 73 ad 16 ac 94 e3 c8 bc ec be b4 b7 a5 5e 1b 49 27 1b 99 32 20 1e 60 58 c0 bf 7e 73 8f f1 c9 c9 f0 61 34 25 6e cb a0 46 b5 23 57 63 4c 82 cf 4c e4 9 5f 29 ec a6 fb 44 e5 51 d1 76 ff ba 56 a2 44 f0 c5 f 29 e5 4c 39 b1 91 54 53 49 95 d7 21 23 64 e1 f8 6c cb 5d 10 e9 5a c4 60 b7 f9 82 8f a0 1f d9 be b7 56 da 7f c8 e2 c7 8d 34 79 1 90 71 26 fa 4 c 67 d9 8b b4 aa 4c a5 91 88 ab 67 74 f6 51 f2 8d 9a bd f1 78 5d 1 b5 97 3 b6 6a 92 83

ALICE ENCRYPTED PAYLOAD: 4d e4 c8 fa 53 82 34 72 be a5 10 82 31 b8 1f ac ee d8 43 3 4b 61 48 f3 ad 92 1a 3a 5 75 64 d5 95 ac f5 3f 49 a7 c2 8f 9f 86 e3 85 2d 6c e5 cd 49 c 7f ed 60 cc d6 0 ea 5b df 66 e0 b6 65 88 f6 20 a7 77 9c f4 5a 1c 83 3a 25 76 1e 66 c4 55 c2 a6 21 ae f4 50 f8 75 a7 ee 6c af b7 96 96 eb c 76 b1 b7 3a 20 bd a0 6d b4 56 eb 71 e6 5c ec d9 d6 4d 75 94 39 91 60 bd 8c ce 1e 1f 4b 28 71 e8 e3 16 24 f5 6c 3a 79 69 7b be 2e 69 5b 9 12 b0 1 6c 10 8e bd d8 2a d8 4b 79 4a f8 3f b5 c7 93 56 3 ab 9d c3 b7 f8 22 f1 f9 65 6b 42 47 39 58 ba 6d 72 74 d9 3e 75 37 ea 8a 4a 8d d3 ca 56 7a 0 b9 e2 c3 c3 c7 81 9 54 c0 f2 ee 84 c7 f b6 30 84 18 23 3e ed 8c 1 2e a6 56 b9 4c 54 27 ce 7 9a 71 12 22 91 e0 36 d6 ce 33 87 da b0 8e 81 24 11 93 5b 8 ff 7 b4 a0 19 43 2f 5a a5 b ff b ce 56 46 27 38 e1 68 db 37 d7 e5 db 6d 7b

CA SIGNATURE OVER ALICES'S INFORMATION (ID, G, P, PUBLIC\_KEY): 43 f2 e0 18 9d 7c 23 63 24 c1 2f 57 ad 16 da a8 b1 40 9a d3 fd 6f 85 82 7a f2 f4 90 60 c4 69 30 ab 41 e3 b d1 a9 82 e1 80 c1 ce de 15 a6 99 7f 85 6d 7c 7c 81 74 59 20 1d 1 81 f4 d2 cf cc 6e ed 8a 7 96 c4 fd a6 90 fd 3d 88 39 1e 11 37 91 cb ef 79 df 0 55 88 62 9a b4 44 33 87 1f 72 d4 f1 72 5c 28 2a 97 7f 66 b0 be 35 71 c2 fa 72 84 66 fc bd dc 65 c3 f0 d2 40 bb e1 76 96 91 51 2e 72 d5 7f ff d9 67 71 70 0 8e 38 8f e0 1f ee c6 a8 44 74 17 bc 9a 3c 35 fc 8e f3 e3 43 d6 e9 2a c7 5a 9e 3b 48 61 fe ba 91 f1 e5 1e 86 fe eb 81 e 5a 4d d1 24 61 8c 2 ae 3e ef 4a 4a 63 e6 36 85 fd 68 b9 16 78 75 73 d6 d a7 c1 4f f4 79 a3 6b c6 3c c7 d4 51 58 34 84 8f fc 7d 3e 22 cf 94 3d 7a c2 79 d0 c8 1a 85 9d 6c 2a 6d c0 51 97 32 e1 fc fc 85 af 1d 63 82 f2 45 c8 6c 2d 9 14 c0

VERIFIED CERTIFICATE SUCCESSFULLY FOR Alice

VERIFIED SIGNED PAYLOAD SUCCESSFULLY FROM Alice  
---------------------------------------------------------------------  
SUCCESSFULLY CONCLUDED STS PROTOCOL

1. Одговор на прашањата

* Праќањето на посебен генератор и циклична група за секоја сесија наместо користење на фиксирани вредности за цела мрежа се должи на превенирање на претпроцесирање на вредности за gx за различни вредности на x и со тоа полесно пресметување на разменетиот клуч.
* Јасно е дека gа, gb се потпишуваат со цел автентицирање на добиената вредност од другата страна (таа е добиена точно од личноста со која се разговара) и превенирање на man-in-the-middle со замена на gb со gt од напаѓач како што е случај со plaintext праќање на вредности во обичниот Diffie-Hellman протокол. Шифрирањето со добиениот таен клуч дава дополнителна сигурност дека пораката е добиена од вистинскиот соговорник и истата е неизменета (ако се пратат само потпишаните вредности, напаѓач може да го побара сертификатот на испраќачот и ги дешифрира со неговиот јавен клуч вредностите).
* Воведувањето на сертификати доаѓа природно од користењето на асиметрични клучеви во протоколот бидејќи за пар клучеви може да се гарантира сопственост само преку индиректно потврдување на сопственоста со авторитет за сертифицирање.

1. Не, поради причините изнесени погоре, man-in-the-middle не е возможен во било која фаза од протоколот, но е подложен на unknown key-share напади во кои иако размената на клуч меѓу Alice и Bob е валидна, некој од нив со интервенција на напаѓач добива впечаток дека дели клуч со друга личност.