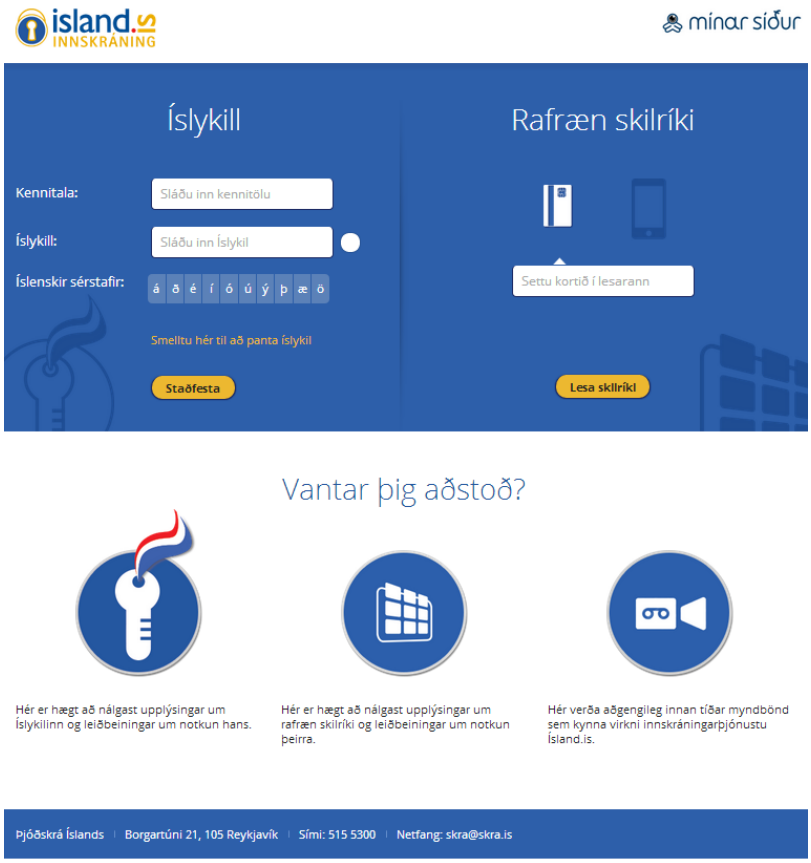
Innskráningarþjónusta Ísland.is

Leiðbeiningar um uppsetningu SAML 2.0



Leiðbeiningar þessar eru skrifaðar fyrir þjónustuveitendur og tæknimenn sem hyggjast innleiða innskráningarþjónustu Ísland.is.

Efnisyfirlit

1 Inngangur 3
1.1 Íslyklar 3
1.2 SAML 2.0 3
1.3 Eldri innskráningarþjónusta Ísland.is 3
2 Yfirlit um uppsetningu og tengingu 4
2.1 Samningur við þjónustuveitanda 4
2.2 Tæknileg atriði 4
2.3 Tenging við innskráningarþjónustu Ísland.is 4
3 Samskipti 6
4 Innihald SAML2.0-skeytis 8
4.1 Kennitala notanda 8
4.2 Nafn notanda 8
4.3 Auðkenning notanda 8
4.4 IP-tala notanda 9
4.5 Kennitala móttakanda 9
4.6 Vottun Íslykils 9
4.7 Kennitala lögaðila 9
4.8 Nafn lögaðila 10
5 Öryggi SAML-skeytis 11
6 Þróunarumhverfi á sérstöku léni 12
6.1 Dæmi 12
Viðaukar 13
1.1 Dæmi um SAML2-skeyti 13
1.2 .Net sýnidæmi 14
1.3 Java sýnidæmi 16
1.4 PHP sýnidæmi 21

1 Inngangur

Innskráningarþjónusta Ísland.is er auðkenningarkerfi sem Þjóðskrá Íslands rekur. Markmið með þjónustunni er að bjóða upp á val um innskráningarleiðir miðað við þarfir þjónustuveitenda og viðskiptavina. Boðið er upp á val um Íslykil, styrktan Íslykil og rafræn skilríki, hvort heldur á (debet)kortum eða SIM kortum í farsíma. Íslyklar eru gefnir út af Þjóðskrá Íslands.

Eldri innskráningarþjónusta Ísland.is byggði á SAML 1.2 en kerfið sem lýst er í þessu skjali byggir á SAML 2.0.

Innleiðing á SAML 2.0 ætti við flestar aðstæður ekki að taka mjög langan tíma þar sem margt er mjög svipað og í SAML 1.2. Núna þarf hins vegar ekki að hafa samband við vefþjónustu og hægt er að vinna úr svarinu strax.

Dæmunum í þessu skjali er ætlað að hjálpa og um að gera að nýta kóða þaðan eftir því sem hægt er.

1.1 Íslyklar

Íslyklar eru gefnir út af Þjóðskrá Íslands. Þjóðskrá íslands er einnig útgefandi pappírsskilríkja, vegabréfa og nafnskírteina. Íslykill er kennitala og lykilorð sem er að lágmarki 10 stafir og blanda af bókstöfum, tölustöfum og táknum. Styrktur Íslykill samanstendur af Íslykli og styrkingu (sex stafa tölu) sem send er í farsíma notanda.

1.2 SAML 2.0

Security Assertion Markup Language 2.0 (SAML 2.0) er staðall sem notaður er til að skiptast á auðkenningar- og heimildargögnum milli mismunandi aðila. SAML 2.0 byggir á XML og notar öryggistóka með staðfestum upplýsingum (e: assertions) til að koma á milli upplýsingum um þann sem á að auðkenna frá auðkenningarþjónustum til þjónustuveitenda.

1.3 Eldri innskráningarþjónusta Ísland.is

Eldri innskráningarþjónusta Ísland.is byggði á SAML1.2. Þar þurfti þjónustuveitandinn að gera vefþjónustukall í svokallað rafrænt þjónustulag Ísland.is til að sækja SAML-skeyti sem innskráningarþjónustan útbjó. Notast var við svokallað „token ID“ til að sækja skeytið.

2 Yfirlit um uppsetningu og tengingu

Fyrirspurnir og umsóknir um tengingu skal senda á netfangið island@island.is. Einnig er hægt að hafa samband við Þjóðskrá Íslands, Borgartúni 21, sími 515 5300. Þjóðskrá Íslands er rekstraraðili Ísland.is.

2.1 Samningur við þjónustuveitanda

Þjóðskrá Íslands gerir skriflegan samning við þjónustuveitendur. Í því skyni þarf stofnunin að fá upplýsingar um eftirfarandi:

a. Nafn þjónustuveitanda, kennitala og heimilisfang
b. Nafn og netfang stjórnunarlegs tengiliðs
c. Nafn og netfang tæknilegs tengiliðs

2.2 Tæknileg atriði

Með umsókninni skulu fylgja eftirtalin atriði:

a. Slóð á örugga síðu (https) sem á að birtast eftir auðkenningu (e. return-url). Dæmi: https://www.stofnun.is/eydublad
b. Lógó þjónustuveitanda verður birt í innskráningarglugganum ásamt lógói Ísland.is. Skila þarf lógóinu á á hvítum grunni, þar sem breiddin er 200 pixlar og hæðin er 60 pixlar

Þjóðskrá Íslands útbýr sérstakt auðkenni (ID) þjónustuveitanda sem oftast er það sama og aðalveffang viðkomandi. Þó geta verið tilvik þar sem þjónustuveitandi er með fleiri en eitt auðkenni.

Dæmi: skra.is, vmst.is

Þjónustuveitandi sem er að skipta úr SAML 1.2 yfir í SAML 2.0 og ætlar að nýta sama ID og áður, þarf að hafa samband við island@island.is til þess að samskiptamátanum fyrir viðkomandi ID sé breytt.

2.3 Tenging við innskráningarþjónustu Ísland.is

a. Þjónustuveitandi byrjar á að framsenda notendur yfir á innskráningarsíðu Ísland.is með sínu auðkenni (ID). Dæmi:

   https://innskraning.island.is/?id=skra.is
   https://innskraning.island.is/?id=vmst.is

b. Ef þjónustuveitandi vill hafa áhrif á hvaða kostir eru í boði við innskráningu þá er bætt aftan við slóðina (GET) &qaa=3 til að bjóða upp á innskráningu með styrktum íslykli eða rafrænum skilríkjum og &qaa=4 til að bjóða aðeins upp á innskráningu með rafrænum skilríkjum.

c. Ef þjónustuveitandi vill senda með beiðni auðkenningarnúmer til að fullvissa sig um að hann hafi átt upprunann af auðkenningarbeiðninni þá er bætt aftan við slóðina

(GET) authId=<id>, þar sem <id> er á GUID formi. Dæmi &authid=5110C405-E94A-4B75-9770-6A4CAB5C7AD4

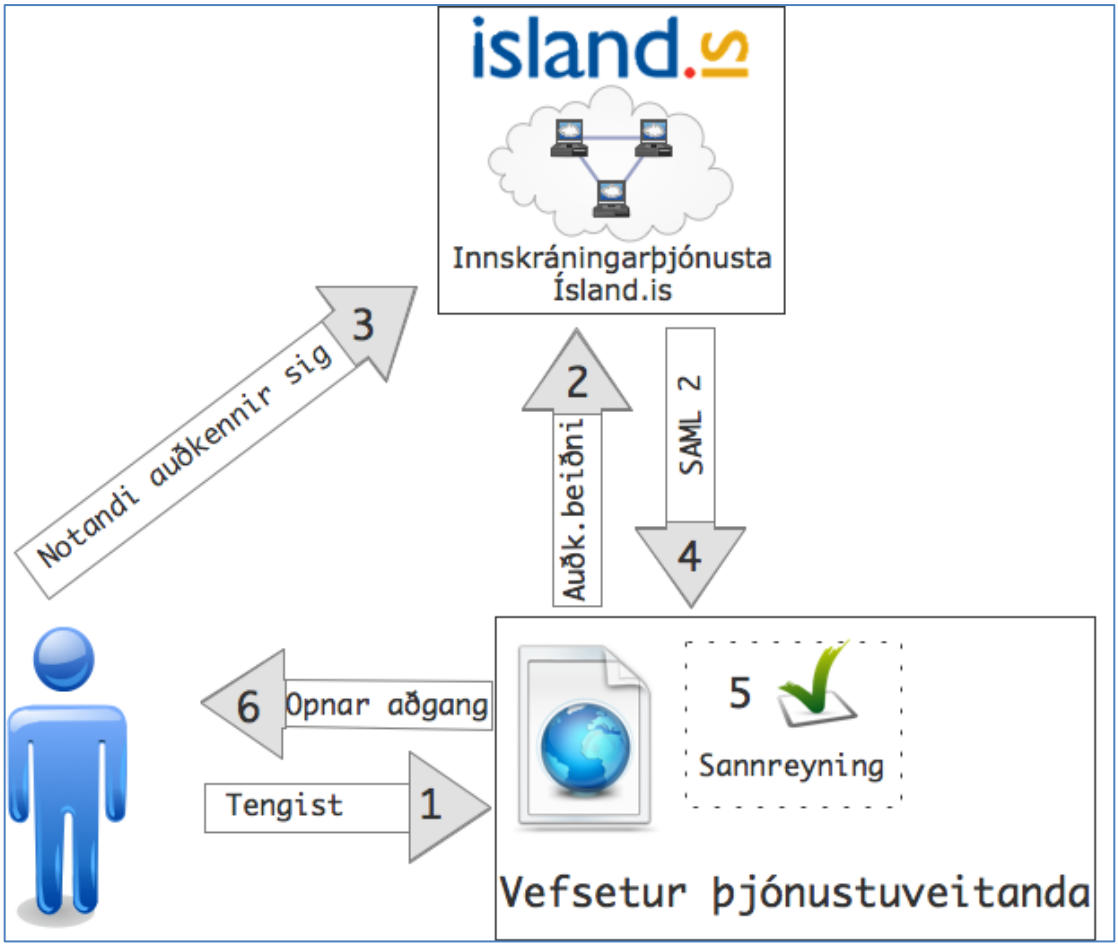
d. Þegar auðkenningu er lokið í innskráningarþjónustu Ísland.is skilar kerfið notendum tilbaka á síðu þjónustuveitanda með rafrænt undirrituðu Base64, UTF-8 kóðuðu SAML2 skeyti (í POST færibreytunni token), sem þjónustuveitandi þarf að lesa úr.

e. Athugið: Það er lykilatriði að þjónustuveitendur standi rétt að því að sannreyna rafrænu undirritunina.

f. Nánari upplýsingar um samskiptin má sjá í næstu köflum.

3 Samskipti

Þessi kafli lýsir í grófum dráttum því hvernig auðkenning fer fram í innskráningarþjónustu Ísland.is.



1. Notandi tengist vefsetri þjónustuveitanda og óskar eftir aðgangi að þeim hluta vefsetursins sem notar innskráningarþjónustu Ísland.is.

2. Þjónustuveitandi metur hve sterkrar auðkenningar er krafist til að nálgast viðkomandi efni/þjónustu á vefsetri þjónustuveitanda. Þjónustuveitandi áframsendir svo notandann yfir á innskráningarþjónustu Ísland.is og tilgreinir í auðkenningarbeiðninni lágmarks styrk þeirra auðkenningar sem krafist er. Boðið er upp á þrjá mismunandi styrkleika auðkenningar, Íslykil, styrktan Íslykil og rafræn skilríki.

3. Notandanum er nú boðið að auðkenna sig með þeim auðkenningarleiðum sem uppfylla lágmarkskröfur sem fram koma í auðkenningarbeiðninni. T.d. ef lágmarkskrafan er rafræn skilríki þá getur notandinn eingöngu auðkennt sig með rafrænum

skilríkjum. Ef lágmarkskrafan er Íslykill þá getur notandinn auðkennt sig annað hvort með Íslykli eða rafrænum skilríkjum.

4. Innskráningarþjónusta Ísland.is auðkennir notandann og sækir nafn hans í þjóðskrá eða fyrirtækjaskrá. Þegar notandinn hefur verið auðkenndur útbýr innskráningarþjónustan SAML2.0 auðkenningartóka (skeyti) sem er rafrænt undirritaður af auðkenningarþjónustunni. Því næst er auðkenningartókinn sendur á vefslóðina sem skilgreind er í innskráningarþjónustunni („return-url“).

5. Vefsetur þjónustuveitanda móttekur SAML 2.0 auðkenningartókann. Vefsetrið sannreynir hvort tókinn sé traustur og rafræn undirritaður af innskráningarþjónustu Ísland.is. Ef AML2.0 tókinn stenst sannreyningu er innihald tókans lesið inn í breytur. SAML tókinn er undirritaður með skilríki Þjóðskrár Íslands sem er gefið út Traustum búnaði. Skilríkjakeðjuna er hægt að nálgast á heimasíðu Auðkennis, http://www.audkenni.is/adstod/skilrikjakedjur.cfm, en hún þarf að vera til staðar í skilríkjageymslu miðlara sem sannreynir tókann. Athugið að keðjan inniheldur þrjú skilríki: Auðkennisrót (rótarskilríki), Traust auðkenni (milliskilríki) og Traustan búnað (milliskilríki) sem þurfa öll að vera til staðar á miðlara sem vinnur SAML skeytið.

6. Byggt á innihaldi SAML2.0 tókans tekur vefsetur þjónustuveitanda ákvörðun um hvort og þá hvaða aðgang viðkomandi notandi á að hafa að vefsetrinu og opnar aðgang í samræmi við það. **Sérstaklega mikilvægt er að staðfesta undirritun og innihald SAML skeytisins við SAML2 innleiðinguna því öryggi innskráningarinnar byggir á að rétt sé unnið úr þessu.**

4 Innihald SAML2.0-skeytis

Eftirfarandi upplýsingar um notanda sem er að tengjast þjónustuveitanda fylgja alltaf í SAML skeyti, undir AttributeStatement:

- Kennitala notanda (UserSSN – Kennitala)
- Nafn notanda (Name – Nafn)
- Auðkenning notanda (Authentication – Auðkenning) á lesanlegu formi, t.d. „Íslykill“ eða „Rafræn skilríki“
- IP tala notanda (IPAddress – IP tala)
- Upplýsingar um vafra notanda (UserAgent – NotandaStrengur)
- Kennitala móttakanda (DestinationSSN – Kennitala móttakanda)

Ef notandi auðkenndi sig með Íslykli þá fylgja einnig með upplýsingar um vottun Íslykils (KeyAuthentication – VottunÍslykils), þ.e. hvernig notandi var auðkenndur þegar sá Íslykill sem nú er í gildi var fyrst búinn til. Ef notandi auðkenndi sig með starfsmannaskilríki þá fylgja einnig með upplýsingar um kennitölu (CompanySSN - KennitalaLögaðila) og nafn (CompanyName - NafnLögaðila) lögaðila.

Í „Attribute statement“ geta stöðluðu breyturnar fengið viss fyrirfram skilgreind gildi. Í þessum kafla verður farið yfir þessi gildi.

4.1 Kennitala notanda

Breytan UserSSN (friendly name „Kennitala“) inniheldur kennitölu notanda sem var að innskrá sig.

4.2 Nafn notanda

Breytan Name (friendly name „Nafn“) inniheldur nafn notanda eins og það kemur fyrir í þjóðskrá eða fyrirtækjaskrá.

4.3 Auðkenning notanda

Breytan Authentication (friendly name „Auðkenning“) inniheldur auðkenningaraðferð notanda. Breytan getur tekið eftirfarandi gildi

- Rafræn skilríki – þegar notandi hefur innskráð sig með rafrænum skilríkjum
- Rafræn starfsmannaskilríki – þegar notandi hefur innskráð sig með rafrænum starfsmannaskilríkjum
- Íslykill – þegar notandi hefur innskráð sig með Íslykli
- Styrktur Íslykill – þegar notandi hefur innskráð sig með Íslykli og styrkingarkóða sendum í síma eða á netfang.
- Styrkt rafræn skilríki – þegar notandi hefur innskráð sig með rafrænum skilríkjum og styrkingarkóða sendum í síma eða á netfang.
- Styrkt rafræn starfsmannaskilríki – þegar notandi hefur innskráð sig með rafrænum starfsmannaskilríkjum og styrkingarkóða sendum í síma eða netfang.
- Óþekkt – þegar innskráning er ekki þekkt (ekki notað – villa)

4.4 IP-tala notanda

Breytan IPAddress (friendly name „IPTala“) inniheldur IP tölu notanda eins og hún birtist við innskráningu. Athugið að notandi getur haft aðra IP tölu gagnvart þjónustuveitanda en innskráningarkerfi svo ekki er hægt að treysta á hún sé sú sama og er í SAML tókanum.

4.5 Upplýsingar um vafra

Breyta UserAgent (friendly name „NotandaStrengur“) inniheldur upplýsingar um vafra notanda sem notaður var við auðkenningu. Dæmi um innihald er „Mozilla/5.0 (Windows NT 6.1; WOW64; rv:26.0) Gecko/20100101 Firefox/26.0“

4.6 Auðkenningarnúmer

Breytan AuthID (friendly name „AuðkenningarNúmer“) inniheldur einkvæmt númer sem þjónustuveitandi hefur sent með auðkenningarbeiðni. Ef þjónustuveitandi sendir ekkert þá birtist ekkert númer í svarinu.

4.7 Kennitala móttakanda

Breytan DestinationSSN (friendly name „KennitalaMóttakanda“) inniheldur kennitölu þjónustuveitanda.

4.8 Vottun Íslykils

Breytan KeyAuthentication (friendly name „VottunÍslykils“) fylgir með þegar notandi hefur innskráð sig með Íslykli eða styrktum Íslykli. Breytan getur tekið eftirfarandi gildi:

- Rafræn skilríki – notandi var innskráður með rafrænum skilríkjum þegar Íslykill var búinn til
- Bréf í pósti – notandi innskráði sig með Íslykli sendum á lögheimili einstaklings þegar Íslykill var búinn til
- Skjal í heimabanka – notandi innskráði sig með Íslykli sendum í heimabankabirtingu einstaklings þegar Íslykill var búinn til
- Afhent hjá Þjóðskrá gegn framvísun löggildra skilríkja
- Afhent hjá samstarfsaðila Þjóðskrár gegn framvísun löggildra skilríkja
- Bréf í pósti á sendiráð og afhending gegn framvísun löggildra skilríkja
- Óþekkt – þegar upplýsingar um það hvað af ofangreindu var notað við gerð Íslykils liggja ekki fyrir

Innihald breytunnar breytist ekki nema notandi fái afhentan nýjan Íslykil með einhverri ofangreindra leiða.

4.9 Kennitala lögaðila

Breytan CompanySSN (friendly name „KennitalaLögaðila“) fylgir með þegar notandi hefur innskráð sig með rafrænum starfsmannaskilríkjum eða styrktum rafrænum starfsmanna-skilríkjum. Kennitala lögaðila er fengin úr starfsmannaskilríki notanda.

4.10 Nafn lögaðila

Breytan CompanyName (friendly name „NafnLögaðila“) fylgir með þegar notandi hefur innskráð sig með rafrænum starfsmannaskilríkjum eða styrktum rafrænum starfsmanna-skilríkjum. Nafn lögaðila er fengin úr starfsmannaskilríki notanda.

5 Öryggi SAML-skeytis

Í SAML skeytinu eru ýmsar upplýsingar um uppruna skeytis og annað sem er nauðsynlegt fyrir rétta uppbyggingu. Þessar upplýsingar eru meðal annars:

- Útgefandi (Issuer)
- Gildistími skeytis (NotBefore og NotAfter eigindi á Assertion/Conditions nóðu)
- Móttakandi (Audience)
- Auðkenning (AuthnContextClassRef nóður undir AuthnStatement/AuthnContext)
- Áfangastaður (Destination í Response header og SubjectConfirmationData)
- IP tala notanda á innskráningarsíðu (Address eigindi á Subject/SubjectConfirmation/SubjectConfirmationData nóðu)
- Upplýsingar um undirritun (Signature)
- Staðfesting á innihaldi (SubjectConfirmationData)
- Upplýsingar um vélbúnað og hugbúnað þess sem auðkenndi sig

SAML skeytið sem innskráningarþjónustan skilar er undirritað með rafrænu skilríki útgefnu af Auðkenni ehf. (Traustur bunadur). Skeytið er varpað með xml-exc-c14n áður en það er hakkað (digest) með SHA256 og undirritað með 2048bita RSA lykli. Til að staðfesta uppruna skeytis verður að athuga að:

- Skeytið sé undirritað með skilríki í eigu Þjóðskrár Íslands (SERIALNUMBER = 6503760649 í „subject“ á skilríki)
- Undirritun sé gild, þ.e. að undirritun sé framkvæmd með skilríkinu hér fyrir ofan
- Skilríki sé gilt, þ.e. það sé gefið út af Auðkenni og hafi ekki verið gert ógilt.
- Skeytið sé í gildi, þ.e. að gildistími sé ekki útrunninn
- Í upphaflegum leiðbeiningum var lagt fyrir þjónustuveitanda að staðfesta að IP tala notanda í skeyti væri sú sama og notanda sem er að auðkenna sig. Í ljós hefur komið að þetta gengur ekki, því á stórum netum á Íslandi er iðulega verið að nota nokkra proxy þjóna og notandi ekki að koma út á sömu IP tölu gagnvart innskráningarþjónustunni og hann er að birta gagnvart þjónustuveitandanum.
- Upplýsingar um vélbúnað og hugbúnað (user agent) þess sem auðkenndi sig stemmi við upplýsingar sem þjónustuveitandi hefur um vélbúnað og hugbúnað þess er að tengjast.
- Móttakandi sé réttur, þ.e. sá þjónustuveitandi sem bað um auðkenninguna

Flest forritunarmál kunna að vinna með SAML skeyti og staðfesta heilleika þeirra, í .Net eru þessi föll í System.Security.Cryptography og í Java er þetta að finna í OpenSAML.

6 Þróunarumhverfi á sérstöku léni

Ef þjónustuveitandi er með þróunarumhverfi á sérstökum vefslóðum (t.d. öðru léni) og vill geta prófað auðkenninguna þar, er einfaldast að óska eftir því að stofnað verði sérstakt viðbótarauðkenni (ID) sem eingöngu verður notað við prófanir.

6.1 Dæmi

Stofnun hefur með þróunarumhverfi á léninu http://development.stofnun.is og vill geta prófað auðkenninguna þar. Stofnað er prófunarauðkennið *d.stofnun.is* og fyrir það er skráð skilaslóðin http://development.stofnun.is/eydublad (sbr. dæmi 1 hér fyrir ofan). Þá er tengill á auðkenningarsíðu: http://innskraning.island.is/?id=d.stofnun.is

Athugið að ekki er gerð krafa um að prófunarauðkenni endurspegli raunverulegt undirlén.

Viðaukar

1.1 Dæmi um SAML2-skeyti

Í SAML2 skeytinu sem innskráningarþjónustan skilar koma fram allar viðeigandi upplýsingar um notanda og uppruna auðkenningar. Upplýsingarnar eru afhentar rafrænt undirritaðar með skilríki innskráningarþjónustunnar. Dæmi um SAML 2 skeyti má sjá hér fyrir neðan.


```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" ID="_1d62ec4e-ef50-4ca0-ad65-805a126a5e99"
Version="2.0" IssueInstant="2014-01-17T15:15:52.1725761Z"
Destination="https://login.advania.is/islogin.aspx"
xmlns="urn:oasis:names:tc:SAML:2.0:protocol"><Issuer
xmlns="urn:oasis:names:tc:SAML:2.0:assertion">Þjóðskrá Íslands</Issuer><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#rsa-sha1" /><Reference URI="#_1d62ec4e-ef50-4ca0-
ad65-805a126a5e99"><Transforms><Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /><Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /></Transforms><DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"
/><DigestValue>uDmYbiiCr3Btq3x2IGvHiXnGeJj0cRCnc9cHreQsFFc=</DigestValue></Reference></SignedI
nfo><SignatureValue>HnFClmDhLmjG/RBeiFiG2XsIB5jhRAAXmAcNKp+uDYnLvwZWYvUAZg3pkOZ9vjy2hpQ8LKY+/A
rXrwL+p6whw8yaHnbvIY9R95GrMhGle5QyDQ9RJB7DvUm4l1Vo/taR4NeB5Y/vpUuaelqeFsz5DY7IzfyfBSkSuydXIPCj
d769uI5Jkpg5O3AhWZ5oS3YjP5bU2ghvgm4lUTckYll5u1OgKGlS/woksl2WrlUcvAw0CFnsz4nlN7HM5DMJ15VMQ96hvh
aGBQiVgf5ZeYXkwFrZjl3P1rVbnp7CQ4pKTsgwj8eE5Canz0ejdeatIpCMdPh1SzbIlIKCmlu2NtrSrg==</SignatureV
alue><KeyInfo><X509Data><X509Certificate>MIIGDDCCBPSgAwIBAgICCbYwDQYJKoZIhvcNAQEFBQAwgZIxCzAJB
gNVBAYTAklTMRMwEQYDVQQFEwo1MjEwMDAyNzkwMRYwFAYDVQQKEw1BdWRrZW5uaSBlaGYuMSMwIQYDVQQLExpVdGdlZmF
uZGkgYnVuYWRhcnNraWxyaWtqYTEWMBQGA1UECxMNTWlsbGlza2lscmlraTEZMBcGA1UEAxMQVHJhdXN0dXIgYnVuYWR1c
jAeFw0xMzA4MTMxMzU2MzFaFw0xNDA4MTMxMzU2MzFaMIHCMQswCQYDVQQGEwJJUzEeMBwGA1UECgwVw55qw7PDsHNrcsO
hIMONc2xhbmRzMRgwFgYDVQQLEw9CdW5hZGFyc2tpbHJpa2kxJTAjBgNVBAsMHFVuZGlycml0dW4gZcOwYSBhdcOwa2Vub
mluZyAxHTAbBgkqhkiG9w0BCQEWDnZlcmtAaXNsYW5kLmlzMRMwEQYDVQQFEwo2NTAzNzYwNjQ5MR4wHAYDVQQDExVJbm5
za3JhbmluZyBJc2xhbmQuaXMwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCkWzKHkw3ckRb55wX9K92Rj4yNv
OIqj9KUYAaO6dmzSRKtZ3APpbsPL+d1f7ObzSyvPZ6hl0H+mgs1l5nsCc4B1E0xR9t7d5AgwF66xYN/zepPfJ95IV3zlmn
nj+CPG4quyJs6HEgyxNH79X9v5lq+xe0WtoHgaY2/vTvceOPOkdksDnCFuD/OvjvVyZ0bTB3Ae0kBNbJRQCAWRpuVSp2I3
xEfXhhqbuZOsv+WUvaOL1januOWYi9d6IoaC8CO3RPRvRcS9q5q/iDu2HYECiqT1ZHXu+u7ZosF0Tn1lt0u//8E6YC051F
+E2XDCtoqiO7QS0sE9Fntxpmbfv0wrlL1AgMBAAGjggI4MIICNDAMBgNVHRMBAf8EAjAAMIIBHAYDVR0gBIIBEzCCAQ8wg
gELBglggmABAgEBBAEwgf0wgcQGCCsGAQUFBwICMIG3GoG0VGhpcyBjZXJ0aWZpY2F0ZSBpcyBpbnRlbmRlZCBmb3IgZGl
naXRhbCBzaWduYXR1cmVzIGFuZCBhdXRoZW50aWNhdGlvbi4gVGhpcyBjZXJ0aWZpY2F0ZSBmdWxmaWxzIHRoZSByZXF1a
XJlbWVudHMgb2Ygbm9ybWFsaXplZCBjZXJ0aWZpY2F0ZSBwb2xpY3kgKE5DUCkgZGVmaW5lZCBpbiBFVFNJIFRTIDEwMiA
wNDIuMDQGCCsGAQUFBwIBFihodHRwOi8vY3AuYXVka2VubmkuaXMvdHJhdXN0dXJidW5hZHVyL2NwMHMGCCsGAQUFBwEBB
GcwZTAjBggrBgEFBQcwAoYXaHR0cDovL29jc3AuYXVka2VubmkuaXMwPgYHYIJgAgFjBoYzaHR0cDovL2NkcC5hdWRrZW5
uaS5pcy9za2lscmlraS90cmF1c3R1cmJ1bmFkdXIucDdiMAsGA1UdDwQEAwIF4DAfBgNVHSMEGDAWgBRv7NsRLAEDaxtgr
rb4aNlAAc2OODBCBgNVHR8EOzA5MDegNaAzhjFodHRwOi8vY3JsLmF1ZGtlbm5pLmlzL3RyYXVzdHVyYnVuYWR1ci9sYXR
lc3QuY3JsMB0GA1UdDgQWBBQPikAxifriUSZrQMrWjxyY2PjEdzANBgkqhkiG9w0BAQUFAAOCAQEABMMaCGABkb40bOrFx
ZIM+YDeYYGne4npdM4QGupkv2aDP2bOhohGw9rCbaxL9lm8BHGHtmWwerwX/3lrtnt367pMc/TRon+vVgnxo9wycmP2u2P
NyXrEI8X297PBJoKlZ/ouDnITVqH/BVvwaPSiYqz78h/kzQRNtFDGLT8O/CqfCRB5qmGNx8I01tcOMq4EHrc/hwHQc1fog
tHJ7JaC5NMO1bg853yl2XC3Dww8OmzAHkCFvKx8aBeEyiesXJHpFj292KohHZhOj1yR5eo9bdqdcjTGkWRn0hN+6vZkdMB
IqhkNS7kyxGpvyniCe3VJZIhaXHIZrbx1aYGlyoCvxg==</X509Certificate></X509Data></KeyInfo></Signatur
e><Status><StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" /></Status><Assertion
Version="2.0" ID="_321427d1-3a6c-4d56-9ada-8d90924f6014" IssueInstant="2014-01-
17T15:15:52.1725761Z" xmlns="urn:oasis:names:tc:SAML:2.0:assertion"><Issuer>Þjóðskrá
Íslands</Issuer><Subject><NameID
NameQualifier="island.is">Innskráningarþjónusta</NameID><SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"><SubjectConfirmationData Address="127.0.0.1"
NotOnOrAfter="2014-01-17T15:25:52.1745763Z" Recipient="http://localhost:63841/GetBack.aspx"
/></SubjectConfirmation></Subject><Conditions NotBefore="2014-01-17T15:15:52.1745763Z"
NotOnOrAfter="2014-01-
17T15:25:52.1745763Z"><AudienceRestriction><Audience>login.advania.is</Audience></AudienceRest
riction></Conditions><AuthnStatement AuthnInstant="2014-01-
17T15:15:52.1725761Z"><SubjectLocality Address="212.30.225.139"
/><AuthnContext><AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:TLSClient</AuthnC
ontextClassRef></AuthnContext></AuthnStatement><AttributeStatement><Attribute Name="UserSSN"
```

```
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="Kennitala"><AttributeValue
xsi:type="xsd:string">1234567890</AttributeValue></Attribute><Attribute Name="Name"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="Nafn"><AttributeValue xsi:type="xsd:string">Jón
Jónsson</AttributeValue></Attribute><Attribute Name="Authentication"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="Auðkenning"><AttributeValue xsi:type="xsd:string">Rafræn
skilríki</AttributeValue></Attribute><Attribute Name="IPAddress"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="IPTala"><AttributeValue
xsi:type="xsd:string">127.0.0.1</AttributeValue></Attribute><Attribute Name="UserAgent"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="NotandaStrengur"><AttributeValue xsi:type="xsd:string">Mozilla/5.0 (Windows NT
6.1; WOW64; rv:26.0) Gecko/20100101 Firefox/26.0</AttributeValue></Attribute><Attribute
Name="AuthID" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="AuðkenningarNúmer"><AttributeValue xsi:type="xsd:string">0807DA8D-3299-4FF4-
BEB2-54727A50FFBD</AttributeValue></Attribute><Attribute Name="DestinationSSN"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
FriendlyName="KennitalaMóttakanda"><AttributeValue
xsi:type="xsd:string">5902697199</AttributeValue></Attribute></AttributeStatement></Assertion>
</Response>.
```

1.2 .Net sýnidæmi

Til að vinna með SAML skeytið í .Net þarf að hlaða því í XmlDocument og þaðan inn í SignedXml. Því næst er hægt að beyta þekktum aðferðum í SignedXml til að sannreyna skeytið. Athugið að tókinn frá innskráningarþjónustunni er Bas64 kóðaður.

```
using System.Security.Cryptography.Xml;
using System.Security.Cryptography.X509Certificates;
using System.Xml;
using System.Xml.Serialization;


public static bool verifySamlSimple(string samlString, string ip,
       string ua, string authId, out string message)
 {
     message = "";
     XmlDocument doc = new XmlDocument();
     doc.PreserveWhitespace = true;
     doc.LoadXml(samlString);

     SignedXml signedXml = new SignedXml(doc);
     //Sækjum undirritun og skoðum
     XmlNodeList nodeList = doc.GetElementsByTagName("Signature");
     XmlNodeList certList = doc.GetElementsByTagName("X509Certificate");
     signedXml.LoadXml((XmlElement)nodeList[0]);
     byte[] certData = Encoding.Default.GetBytes(certList[0].InnerText);
     X509Certificate2 cert = new X509Certificate2(certData);

     //Hér er mikilvægt að setja ekki true í seinna gildi nema útfærð sé
     //aðgerð sem sannreynir gildi skilríkis sérstaklega
     bool signatureOk = signedXml.CheckSignature(cert, false);
     if (signatureOk)
         message = "Signature OK. ";
     else
```

```
                message = "Signature not OK. ";

            bool certOk = false;
            if (cert.Issuer.StartsWith("CN=Traustur bunadur") &&
            cert.Subject.Contains("SERIALNUMBER=6503760649"))
            {
                certOk = true;
                message += "Certificate is OK. ";
            }
            else
                message += "Certificate not OK. ";


            DateTime nowTime = DateTime.UtcNow;
            //Sækjum tíma í Conditions og berum saman
            XmlNodeList condNodeList = doc.GetElementsByTagName("Conditions");
            DateTime fromTime =
DateTime.Parse(condNodeList[0].Attributes["NotBefore"].Value);
            DateTime toTime =
DateTime.Parse(condNodeList[0].Attributes["NotOnOrAfter"].Value);

            bool timeOk = false;
            if (nowTime > fromTime && toTime > nowTime)
            {
                timeOk = true;
                message += "SAML time valid. ";
            }
            else if (nowTime < fromTime)
                message += "From time has not passed yet. ";
            else if (nowTime > toTime)
                message += "To time has passed. ";

            //Skoðum nú IP tölu notanda, notandastreng og auth id úr Attributes
            bool ipOk = false;
            bool uaOk = false;
            bool authidOk = false;
            XmlNodeList attrList =
            doc.GetElementsByTagName("Attribute");
            if (attrList.Count > 0)
            {
                foreach (XmlNode attrNode in attrList)
                {
                    XmlAttributeCollection attrCol = attrNode.Attributes;
                    //Staðfestum að IP tala sé sú sama - athugið að ekki er
                    //alltaf hægt að treysta á að IP tala sé sú sama
                    if (attrCol["Name"].Value.Equals("IPAddress"))
                    {
                        ipOk = attrNode.FirstChild.InnerText.Equals(ip);
                        message += "Correct client IP.  ";
                    }
                    //Staðfestum að user agent strengur sé sá sami
                    if (attrCol["Name"].Value.Equals("UserAgent"))
                    {
                        uaOk = attrNode.FirstChild.InnerText.Equals(ua);
                        message += "Correct client user agent. ";
                    }
                    //Staðfestum að auðkenningarnúmer sé það sama
```

```
                    if (attrCol["Name"].Value.Equals("AuthID"))
                    {
                        authidOk =
attrNode.FirstChild.InnerText.Equals(authId);
                        message += "Correct client auth ID. ";
                    }
                }
                if (!ipOk)
                    message += "Incorrect client IP. ";
                if (!uaOk)
                    message += "Incorrect client user agent. ";
                if (!uaOk)
                    message += "Incorrect auth ID. ";
            }
            else
                message += "No Attributes found";

            // Skeytið er í lagi ef undirritun, skilríki, tímar eru í lagi
            // ásamt ip-tölu, notandastreng eða auðkenningarnúmer.
            return signatureOk && certOk && timeOk &&
              (ipOk || uaOk || authidOk);
        }
```

1.3 Java sýnidæmi

Til að vinna með SAML skeytið í Java þarf að hlaða því í SignableSAMLObject með unmarshalling. Eftir það er hægt að staðfesta heilleika skilríkis o.fl.

```
KeyStore trustStore;

public boolean validateSaml(final String samlString, final String userIP,
      final String userAgent, final String authId)
{
  boolean ok = false;

  try {
    SignableSAMLObject signedObject =
      (SignableSAMLObject)this.unmarshall(samlString);

    if (signedObject != null)
    {
      SignableSAMLObject samlObject =
              (SignableSAMLObject)this.validateSignature(signedObject,
trustStore);
      if (samlObject != null)
      {
        Assertion assertion = this.getAssertion((Response)samlObject,
userIP,
          false);
        if(assertion!=null){
          final DateTime serverDate = new DateTime();

          if
(assertion.getConditions().getNotBefore().isAfter(serverDate))
          {
            throw new Exception("Token date valid yet (getNotBefore =  " +
```

```
                assertion.getConditions().getNotBefore()

                + " ), server_date: " + serverDate);
            }
            if
 (assertion.getConditions().getNotOnOrAfter().isBefore(serverDate)) {
              throw new Exception("Token date expired (getNotOnOrAfter =  "
              + assertion.getConditions().getNotOnOrAfter()
              + " ), server_date: " + serverDate);
            }
            // Validate the assertions for IP, useragent and authId.
            validateAssertion(assertion, userIP, userAgent, authId);

            ok = true;
          }
        }
      }
    } catch (Exception e) {
      //SAML not verified
      e.printStackTrace();
    }

    return ok;
}

//Unmarshall SAML string
private final XMLObject unmarshall(final String samlString) throws
Exception {
    try {
       byte[] samlToken = Base64.base64ToByteArray(samlString);
       final BasicParserPool ppMgr = new BasicParserPool();
       final HashMap<String, Boolean> features = new HashMap<String,
 Boolean>();
       features.put(XMLConstants.FEATURE_SECURE_PROCESSING, Boolean.TRUE);
       ppMgr.setBuilderFeatures(features);
       ppMgr.setNamespaceAware(true);

       Document document = ppMgr.parse(new ByteArrayInputStream(samlToken));
       if (document != null){
        final Element root = document.getDocumentElement();
        final UnmarshallerFactory unmarshallerFact =
         Configuration.getUnmarshallerFactory();
        if (unmarshallerFact != null && root != null)
        {
           final Unmarshaller unmarshaller =
 unmarshallerFact.getUnmarshaller(root);
           try {
             return unmarshaller.unmarshall(root);
           } catch (NullPointerException e){
             throw new Exception("NullPointerException", e);
           }
        } else {
           throw new Exception("NullPointerException : unmarshallerFact or
 root is
             null");
        }
       } else {
         throw new Exception("NullPointerException : document is null");
       }
     } catch (XMLParserException e) {
       throw new Exception(e);
     } catch (UnmarshallingException e) {
```

```
      throw new Exception(e);
    } catch (NullPointerException e) {
      throw new Exception(e);
    }
  }

  private final SAMLObject validateSignature(final SignableSAMLObject
        tokenSaml, KeyStore keyStore) throws Exception {
    try {
      // Validate structure signature
      final SAMLSignatureProfileValidator sigProfValidator =
      new SAMLSignatureProfileValidator();
      try {
        // Indicates signature id conform to SAML Signature profile
        sigProfValidator.validate(tokenSaml.getSignature());
      } catch (ValidationException e) {
        //ValidationException: signature isn't conform to SAML Signature
  profile.
        throw new Exception(e);
      }

      String aliasCert = null;
      X509Certificate certificate;

      final KeyInfo keyInfo = tokenSaml.getSignature().getKeyInfo();

      final org.opensaml.xml.signature.X509Certificate xmlCert = keyInfo
          .getX509Datas().get(0).getX509Certificates().get(0);

      final CertificateFactory certFact = CertificateFactory
          .getInstance("X.509");
      final ByteArrayInputStream bis = new ByteArrayInputStream(Base64.
          base64ToByteArray(xmlCert.getValue()));
      final X509Certificate cert = (X509Certificate) certFact
          .generateCertificate(bis);

      // Exist only one certificate
      final BasicX509Credential entityX509Cred = new BasicX509Credential();
      entityX509Cred.setEntityCertificate(cert);

      try {
        cert.checkValidity();
      }    catch (CertificateExpiredException exp) {
        throw new Exception("Certificate expired.");
      }    catch (CertificateNotYetValidException exp) {
          throw new Exception("Certificate not yet valid.");
      }

      boolean trusted = false;

      for (final Enumeration<String> e = keyStore.aliases();
  e.hasMoreElements();)
      {
        aliasCert = e.nextElement();
        certificate = (X509Certificate) keyStore.getCertificate(aliasCert);

        try {
          cert.verify(certificate.getPublicKey());
          trusted = true;
          break;
        }    catch (Exception ex) {
          //Do nothing - cert not trusted yet
```

```
        }
      }

      if (!trusted)
        throw new Exception("Certificate is not trusted.");
      else {
        if
(cert.getSubjectDN().toString().contains("SERIALNUMBER=6503760649")
            && cert.getIssuerDN().toString().startsWith("CN=Traustur
bunadur"))
          trusted = true;
        else {
          throw new Exception("Certificate is not trusted.");
        }
      }

      // Validate signature
      final SignatureValidator sigValidator = new SignatureValidator(
          entityX509Cred);
      sigValidator.validate(tokenSaml.getSignature());

    } catch (Exception e) {
      e.printStackTrace();
  }
  return tokenSaml;
}

private Assertion getAssertion(final Response samlResponse,
       final String userIP, final boolean ipValidate) throws Exception {
  if (samlResponse.getAssertions() == null
      || samlResponse.getAssertions().isEmpty()) {
    //Assertion is null or empty
    return null;
  }

  final Assertion assertion = (Assertion)
samlResponse.getAssertions().get(0);

  for (final Iterator<SubjectConfirmation> iter = assertion.getSubject()
      .getSubjectConfirmations().iterator(); iter.hasNext();) {
    final SubjectConfirmation element = iter.next();
    final boolean isBearer = SubjectConfirmation.METHOD_BEARER
        .equals(element.getMethod());

    if (ipValidate) {
      if (isBearer) {
        if (StringUtils.isBlank(userIP)) {
          throw new Exception("browser_ip is null or empty.");
        } else if (StringUtils.isBlank(element
            .getSubjectConfirmationData().getAddress())) {
          throw new Exception("token_ip attribute is null or empty.");
        }
      }
      final boolean ipEqual = element.getSubjectConfirmationData()
          .getAddress().equals(userIP);

      // Validation ipUser
      if (!ipEqual && ipValidate) {
        throw new Exception("IPs doesn't match : token_ip ("
          + element.getSubjectConfirmationData().getAddress()
          + ") browser_ip ("+ userIP + ")");
      }
```

```
    }
  }
  return assertion;
}

/**
 * Validate assertions for IP, user agent and auth ID
 * @param assertion The assertion to validate
 * @param ip The user IP
 * @param ua The users user agent
 * @param authId The auth ID
 * @throws Exception
 */
private void validateAssertion(final Assertion assertion, String ip,
       String ua, String authId ) throws Exception {
       final List<XMLObject> listExtensions =
              assertion.getOrderedChildren();

       boolean find = false;
       AttributeStatement requestedAttr = null;

       // Search the attribute statement.
       for (int i = 0; i < listExtensions.size() && !find; i++) {
              final XMLObject xml = listExtensions.get(i);
              if (xml instanceof AttributeStatement) {
                     requestedAttr = (AttributeStatement) xml;
                     find = true;
              }
       }

       if (!find) {
              throw new Exception (
              "AttributeStatement it's not present.");
       }

       final List<Attribute> reqAttrs = requestedAttr.getAttributes();

       String attributeName, tempValue;
       XMLObject xmlObj;
       boolean ipOk = false, uaOk = false, authIdOk = false;

       // Process the attributes.
       for (int nextAttribute = 0; nextAttribute < reqAttrs.size();
              nextAttribute++) {
              final Attribute attribute = reqAttrs.get(nextAttribute);

              attributeName = attribute.getName();
              if (attributeName.equals("IPAddress"))
              {
                     xmlObj = attribute.getOrderedChildren().get(0);
                     tempValue = ((XSStringImpl) xmlObj).getValue();
                     ipOk = tempValue.equals(ip);
              }
              if (attributeName.equals("UserAgent"))
              {
                     xmlObj = attribute.getOrderedChildren().get(0);
                     tempValue = ((XSStringImpl) xmlObj).getValue();
                     uaOk = tempValue.equals(ua);
              }
              if (attributeName.equals("AuthID"))
              {
                     xmlObj = attribute.getOrderedChildren().get(0);
```

```
                    tempValue = ((XSStringImpl) xmlObj).getValue();
                    authIdOk = tempValue.equals(ip);
                }
        }
        if (ipOk || authIdOk || uaOk)
                System.out.println("Assertion valid.");
        else
                throw new Exception
                (String.format("Assertions not valid. IP valid %b, "
                        + "user agent" valid %b, auth ID valid %b",
                        ipOk, uaOk, authIdOk));
}
```

1.4 PHP sýnidæmi

Til að sannreyna SAML í PHP er notast við xmlseclibs (https://code.google.com/p/xmlseclibs/).

```
        function verifySaml()
        {
                include 'xmlseclibs.php';
                $token = $_POST["token"];
                if ($token != NULL)
                {
                    $xmlDoc = new DOMDocument();
                    $saml = base64_decode($token);
                    $xmlDoc->loadXML($saml);
                    $xmlsec = new XMLSecurityDSig();
                    $objXMLSecDSig = new XMLSecurityDSig();

                    $objDSig = $objXMLSecDSig->locateSignature($xmlDoc);

                    if ($objDSig == NULL) {
                            throw new Exception("Cannot locate Signature Node");
                    }
                    $objXMLSecDSig->canonicalizeSignedInfo();
                    $objXMLSecDSig->idKeys = array('ID');
                    $objXMLSecDSig->idNS          =          array('wsu'=>'http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd');
                    $retVal = $objXMLSecDSig->validateReference();

                    if ($retVal == NULL) {
                            throw new Exception("Reference Validation Failed");
                    }

                    if (!VerifyDate($xmlDoc))
                    {
                        throw new Exception("Conditions not valid.");
                    }

                    //  Ekki treystandi
                    /*if (!verifyConditions($xmlDoc, get_client_ip()))
                    {
                        throw new Exception("Invalid client ip.");
                    }*/

                    $objKey = $objXMLSecDSig->locateKey();
                    if (! $objKey ) {
                            throw new Exception("Key not found");
                    }
                    $key = NULL;

                    $objKeyInfo = XMLSecEnc::staticLocateKeyInfo($objKey, $objDSig);
```

```
                    if (!verifyCert($objKeyInfo))
                    {
                        throw new Exception("Certificate not valid");
                    }

                    if (! $objKeyInfo->key && empty($key)) {
                            $objKey->loadKey(dirname(__FILE__) . '/mycert.pem', TRUE);
                    }

                    if (!$objXMLSecDSig->verify($objKey)) {
                            throw new Exception("Signature invalid!");
                    }
                    else {
                         checkUserAgent($xmlDoc, get_user_agent());
                        checkIP($xmlDoc, get_client_ip());
                        checkAuthID($xmlDoc, get_auth_id());
                        return "Signature valid.";
                    }
                }
            }

            function locateConditions($doc)
            {
                $xpath = new DOMXPath($doc);
                $xpath->registerNamespace('assertion',
'urn:oasis:names:tc:SAML:2.0:assertion');
                $nodeset = $xpath->query(".//assertion:Assertion/assertion:Conditions", $doc);
                return $nodeset->item(0);
            }

            function locateSubjectConfirmation($doc)
            {
                $xpath = new DOMXPath($doc);
                $xpath->registerNamespace('assertion',
'urn:oasis:names:tc:SAML:2.0:assertion');
                $nodeset                                =                                $xpath-
>query(".//assertion:Assertion/assertion:Subject/assertion:SubjectConfirmation/assertion:Subje
ctConfirmationData", $doc);
                return $nodeset->item(0);
            }

            function verifyConditions($doc, $ip)
            {
                $conditions = locateSubjectConfirmation($doc);
                if (!$conditions)
                {
                        throw new Exception("Unable to locate Conditions");
                        return false;
                }

                try
                {
                        $address = $conditions->getAttribute('Address');
                }
                catch (Exception $e)
                {
                    throw new Exception("Exception while locating address");
                    return false;
                }

                if (!strcmp($ip, "::1"))
                    $ip = "127.0.0.1";

                if (strcmp($address, $ip))
                {
                    throw new Exception("Invalid IP address.");
                    return false;
```

```
                }
                else
                    return true;

            }

            function VerifyDate($doc)
            {
                try
                {
                        $conditions = locateConditions($doc);
                }
                catch (Exception $e)
                {
                    throw new Exception("Exception while locating Conditions");
                    return false;
                }

                if (!$conditions)
                {
                        throw new Exception("Unable to locate Conditions");
                        return false;
                }

                try
                {
                        $start = $conditions->getAttribute('NotBefore');
                        $end = $conditions->getAttribute('NotOnOrAfter');
                }
                catch (Exception $e)
                {
                    throw new Exception("Exception while locating start or end");
                    return false;
                }

                if (!$start || !$end)
                {
                        throw new Exception("Unable to locate start (NotBefore) or end
(NotOnOrAfter)");
                        return false;
                }

                date_default_timezone_set('Atlantic/Reykjavik');
                $startTime = strtotime($start);
                $endTime = strtotime($end);

                if (!is_int($startTime) || !is_int($endTime))
                {
                        throw new Exception("Unable to get time from start (NotBefore) or end
(NotOnOrAfter)");
                        return false;
                }

                $now = date(time());

                $inSpan = $startTime < $now && $now < $endTime;

                if (!$inSpan)
                {
                        throw new Exception("Response is not within specified timeframe");
                        return false;
                }

                return true;
            }

            function verifyCert($objKeyInfo)
            {
```

```
    $caFile = file_get_contents("TrausturBunadur.pem");
    $caCert = openssl_x509_read($caFile);
    $caCertParsed = openssl_x509_parse($caCert, true);
    $parsed = openssl_x509_parse($objKeyInfo->getX509Certificate());

    date_default_timezone_set('Atlantic/Reykjavik');
    $dateFrom = date($parsed['validFrom_time_t']);
    $dateTo = date($parsed['validTo_time_t']);
    $nowTime = date(time());
    if ($nowTime < $dateFrom || $nowTime > $dateTo)
    {
        throw new Exception("Certificate expired or not valid yet");
    }

    $kennitala = $parsed['subject']['serialNumber'];
    $issuer = $parsed['issuer']['CN'];

    if ($kennitala != "6503760649")
    {
        throw new Exception("Ekki rétt kennitala í undirritunarskilríki");
        return false;
    }

    if ($issuer!= "Traustur bunadur")
    {
        throw new Exception("Ekki réttur útgefandi undirritunarskilríkis");
        return false;
    }

    $subjectKey = $caCertParsed['extensions']['subjectKeyIdentifier'];
    $authKey = $parsed['extensions']['authorityKeyIdentifier'];
    $authKey = str_replace('keyid:', '', $authKey);
    if (!strcasecmp($subjectKey,$authKey))
    {
        throw new Exception("Not correct CA");
        return false;
    }

    return true;
}

//Athugið að treysta ekki eingöngu þessu prófi
function checkIP($xmlDoc, $ip)
{
    if ($xmlDoc != NULL)
    {
        if (!strcmp($ip, "::1"))
            $ip = "127.0.0.1";
        $searchNode = $xmlDoc->getElementsByTagName( "Attribute" );

        foreach( $searchNode as $attribute )
        {
            $friendly = $attribute->getAttribute("FriendlyName");
            if ($friendly == "IPTala")
            {
                if ($attribute->nodeValue == $ip)
                    echo "IP OK <br>";
                else
                    echo "IP not OK <br>";
                break;
            }
        }
    }
}

function checkUserAgent($xmlDoc, $ua)
{
    if ($xmlDoc != NULL)
```

```
    {
        $searchNode = $xmlDoc->getElementsByTagName( "Attribute" );

        foreach( $searchNode as $attribute )
        {
            $friendly = $attribute->getAttribute("FriendlyName");
            if ($friendly == "NotandaStrengur")
            {
                if ($attribute->nodeValue == $ua)
                    echo "User agent OK <br>";
                else
                    echo "User agent not OK <br>";
                break;
            }
        }
    }
}

function checkAuthID($xmlDoc, $authid)
{
    if ($xmlDoc != NULL)
    {
        $searchNode = $xmlDoc->getElementsByTagName( "Attribute" );

        foreach( $searchNode as $attribute )
        {
            $friendly = $attribute->getAttribute("FriendlyName");
            if ($friendly == "AuðkenningarNúmer")
            {
                if ($attribute->nodeValue == $authid)
                    echo "Auth ID OK <br>";
                else
                    echo "Auth ID not OK <br>";
                break;
            }
        }
    }
}

function get_client_ip() {
    $ipaddress = '';
    if (!empty($_SERVER['HTTP_CLIENT_IP']))
        $ipaddress = $_SERVER['HTTP_CLIENT_IP'];
    else if(!empty($_SERVER['HTTP_X_FORWARDED_FOR']))
        $ipaddress = $_SERVER['HTTP_X_FORWARDED_FOR'];
    else if(!empty($_SERVER['HTTP_X_FORWARDED']))
        $ipaddress = $_SERVER['HTTP_X_FORWARDED'];
    else if(!empty($_SERVER['HTTP_FORWARDED_FOR']))
        $ipaddress = $_SERVER['HTTP_FORWARDED_FOR'];
    else if(!empty($_SERVER['HTTP_FORWARDED']))
       $ipaddress = $_SERVER['HTTP_FORWARDED'];
    else if(!empty($_SERVER['REMOTE_ADDR']))
        $ipaddress = $_SERVER['REMOTE_ADDR'];
    else
        $ipaddress = 'UNKNOWN';

    return $ipaddress;
}

function get_user_agent()
{
    $useragent = '';
    if (!empty($_SERVER['HTTP_USER_AGENT']))
        $useragent = $_SERVER['HTTP_USER_AGENT'];
    return $useragent;
}
```

```
//Þetta þarf að útfæra sérstaklega ef notað er authId
function get_auth_id()
{
    return "1234567890";
}
```