HashEye

// SMART CONTRACT SECURITY AUDIT

Ockam

Security assessment by HashEye · prepared for Ockam

HASHEYE AUDITED

PROJECT	Ockam
CLIENT	Ockam
CATEGORY	Blockchain
PUBLISHED	November 1, 2023
REPORT ID	research-ockam-2023-11-01-13q6ms

This report was produced under HashEye's layered review process — **automated detection**, **pattern correlation**, and **senior manual verification** — with every finding signed off by a human reviewer. Full findings detail and on-chain attestation are available on the report page at hasheye.io/audits/research-ockam-2023-11-01-13q6ms.

Ockam DesignReview November22,2023 Preparedfor: MrinalWadhwa OckamInc.
Preparedby:ScottArciszewski,FredrikDahlgren,MarcIlunga,andJoopvandePol

Abouthasheye Foundedin2012andheadquarteredinNewYork,hasheyeprovidestechnicalsecurity assessmentandadvisoryservicestosomeoftheworld'smosttargetedorganizations.We combinehigh-endsecurityresearchwithareal -worldattackermentalitytoreduceriskand fortifycode.With100+employeesaroundtheglobe,we'vehelpedsecurecriticalsoftware elementsthatsupportbillionsofendusers,includingKubernetesandtheLinuxkernel. Wemaintainanexhaustivelistofpublicationsathttps://github.com/hasheye-io/publications, withlinkstopapers,presentations,publicauditreports,andpodcastappearances. Inrecentyears,hasheyeconsultantshaveshowcasedcutting-edgeresearchthrough presentationsatCanSecWest,HCSS,Devcon,EmpireHacking,GrrCon,LangSec,NorthSec, theO'ReillySecurityConference,PyCon,REcon,SecurityBSides,andSummerCon. Wespecializeinsoftwaretestingandcodereviewprojects,supportingclientorganizations inthetechnology,defense,andfinanceindustries,aswellasgovernmententities.Notable clientsincludeHashiCorp,Google,Microsoft,WesternDigital,andZoom. hasheyealsooperatesacenterofexcellencewithregardtoblockchainsecurity.Notable projectsincludeauditsofAlgorand,BitcoinSV,Chainlink,Compound,Ethereum2.0, MakerDAO,Matic,Uniswap,Web3,andZcash. Tokeepuptodatewithourlatestnewsandannouncements,pleasefollowhasheyeon Twitterandexploreourpublicrepositoriesathttps://github.com/hasheye-io.Toengageus directly,visitour"Contact"pageathttps://www.hasheye.io/contact,oremailusat info@hasheye.io.

ProjectSummary ContactInformation Thefollowingprojectmanagerwasassociatedwiththisproject: SamGreenup,ProjectManager sam.greenup@hasheye.io Thefollowingengineeringdirectorwasassociatedwiththisproject: JimMiller,EngineeringDirector,Cryptography james.miller@hasheye.io Thefollowingconsultantswereassociatedwiththisproject: ScottArciszewski,ConsultantFredrikDahlgren,Consultant

scott.arciszewski@hasheye.iofredrik.dahlgren@hasheye.io
MarcIlunga,ConsultantJoopvandePol,Consultant marc.ilunga@hasheye.iojoop.vandepol@hasheye.io
ProjectTimeline Thesignificanteventsandmilestonesoftheprojectarelistedbelow. DateEvent
September28,2023Pre-projectkickoffcall October10,2023Statusupdatemeeting#1
October16,2023Statusupdatemeeting#2 October20,2023Statusupdatemeeting#3
November1,2023Statusupdatemeeting#4 November6,2023Deliveryofreportdraft
November6,2023Reportreadoutmeeting November22,2023Deliveryofcomprehensivereport hasheye
4OckamDesignReview PUBLIC

ExecutiveSummary EngagementOverview
OckamengagedhasheyetoreviewthesecurityofOckam,whichisasetofprotocolsand
managedinfrastructure.Ockam’sprotocolsaimtoenablesecureend-to-end
communicationbetweenendpointsacrossvarioustopologiesasiftheywereconnected
locally,withoutanymodificationoftheendpointsthemselves.Moreover,usersmaydeploy
theprotocolsontheirpremises,therebynotrelyingonOckam’smanagedinfrastructure
andobviatingtheneedtotrustOckamoritsinfrastructure.
AteamoffourconsultantsconductedthereviewfromOctober2toNovember3,2023,for atotalof11engineer-
weeksofeffort.Ourtestingeffortsfocusedonthesecurityof
Ockam’sprotocolsinthecontextoftwospecificusecases:TCPPortalsandKafkaPortals.
Withfullaccesstodesigndocumentation,wereviewedthedesignoftheprotocols,
focusingonthetwousecasesinscope.Weconductedthereviewusingautomatedand
manualprocesses.Whilewehadaccesstothecurrentimplementationof Ockam’s
protocolstoaidinunderstandingthedesign,theimplementationitselfwasnotinscopefor thereview.
ObservationsandImpact
Ockam’sprotocolsuserobustcryptographicprimitivesaccordingtoindustrybestpractices.
Noneoftheidentifiedissuesposeanimmediaterisktotheconfidentialityandintegrityof
datahandledbythesysteminthecontextofthetwoin-scopeusecases.Themajorityof
identifiedissuesrelatetoinformationthatshouldbeaddedtothedesigndocumentation,
suchasthreatmodeldetails(TOB-OCK-5,andTOB-OCK-1)andincreasedspecificationfor certainaspects(TOB-
OCK-2,TOB-OCK-4,andTOB-OCK-6). Regularquestion-and-
answersessionswereheldduringtheengagementtoclarifyany
questionsabouttheprotocolsandtoprovideinformationthatwasmissingfromthedesign
documentation.Thesewereveryhelpfultounderstandingtheoverallsystemaswellasany
designconsiderationsmadebythedevelopers. Recommendations
Basedonthecodebasematurityevaluationandfindingsidentifiedduringthesecurity
review,hasheyerecommendsthatOckamtakethefollowingsteps: •
Remediatethefindingsdisclosedinthisreport.Althoughnofindingsindicate anyimmediateorhigh-
severityrisktotheOckamsystem,thefindingsshouldbe
addressedaspartofadirectremediationoraspartofanyredesignthatmayoccur
whenaddressingotherrecommendations. hasheye 5OckamDesignReview PUBLIC

• IncorporateallrelevantinformationfromtheQ&Asessionsintothedesign
documentation.Thequestionsaskedduringthisdesignreviewcorrespondto
potentialsystemissuesthatarenotexcludedbythecurrentdesigndocumentation.
Whiletheanswersshowedthattheactualsystemdoesnotsufferfromthese
specificissues,incorporatingtheanswersintothedesigndocumentationwould
helpensurethatthisbecomesanexplicit,ratherthanimplicit,partofthedesign. •
ConsiderasecurityauditoftheimplementationofOckam’sprotocols.
Ultimately,theimplementationofaprotocoldeterminestheriskassociatedwiththe
protocol,asasecuredesigndoesnotimplyasecureimplementation.Issuesinthe
deploymentofaprotocolmayarisefromdiscrepanciesbetweenthedesignandthe
implementationorfromspecificimplementationchoicesthatviolatethe
assumptionsinthedesign.AreviewoftheimplementationofOckam’sprotocolscan
uncoversuchissues,helpingthedeploymentofOckamtobeassafeaspossible.For
specificusecases,itmaybenecessarytoauditthecorrespondingconfigurationof third-
partyinfrastructure,suchasGitHubandAWS. •
ConsiderfurtherformalanalysisofOckam’sprotocols.Inourreview,weused
automatedtoolingtoassistwiththeanalysisintheallottedtime.Formalmodels
provideahighlevelofassuranceinthetargetedprotocolandarealsoagood
startingpointwhentryingtounderstandhowasmallchangeorupdatewould
impacttheoverallsecurityguaranteesprovidedbytheprotocol.
Moreover,ifthethreatmodelandexpectedsecurityguaranteesaredocumentedas
recommendedinthisreport,aformalanalysismaybevaluableinunderstanding
theguaranteesprovidedbythesystem.Forthisreason,werecommendthatOckam
consideradditionalformalanalysesofitsprotocols.ThesectiononCoverage

Limitationsprovidessomeexampleprotocolaspectsthatcouldbenefitfromformal modeling. hasheye 6OckamDesignReview PUBLIC

FindingSeveritiesandCategories Thefollowingtablesprovidethenumberoffindingsbyseverity,difficulty,andcategory. EXPOSUREANALYSIS SeverityCount High0 Medium3 Low0 Informational3 Undetermined0 CATEGORYBREAKDOWN CategoryCount Cryptography6 DifficultyCount Low0 Medium0 High4 Undetermined0 NotApplicable2 hasheye 7OckamDesignReview PUBLIC

ProjectGoals Theengagementwasscopedtoprovideasecurityassessmentof Ockam’sprotocolswith respecttotheTCPPortalsandKafkaPortalsusecases.Specifically,wesoughttoanswer thefollowingnon-exhaustivelistofquestions: • Arethereanycriticaldesignflawsthatcouldresultinthecompromiseofdata confidentialityorintegrity? • Isthedesignvulnerabletoanyknowncryptographicattacks? • Doesthesystemuseanyweakcryptographicprimitives? • Isthetrustinfrastructuresufficientlysecure? • Doesthesystemfollowbestpracticesforthegeneration,distribution,storage,and destructionofcryptographickeys? • Arehandshakesconductedinacryptographicallysecuremanner? • Aresecurechannelprotocolsusedeffectively? • Areaccesscontrols,policies,andcredentialscryptographicallysecure? • IstheNoiseframeworksecurelyusedwithinthedesign? • Doesthedesignofthesystemlenditselfwelltoimprovementsanditerationsover time? • Withinthedesign,isend-to-endencryptionguaranteed? • Doesthedesignenablethedeploymentofthesystemandonboardingofnew clientsinasecuremanner? hasheye 8OckamDesignReview PUBLIC

ProjectTargets Theengagementinvolvedareviewandtestingofthefollowingtarget. Ockamdesigndocumentation Repositoryhttps://app.gitbook.com/o/bSIWJRTRgnhbW4jNMcHT/ Version 2023-11-03 TypeDocumentation hasheye 9OckamDesignReview PUBLIC

ProjectCoverage Thissectionprovidesanoverviewoftheanalysiscoverageofthereview,asdeterminedby ourhigh-levelengagementgoals.Ourapproachesincludedthefollowing: • Manualreviewofthedesigndocumentation.Thereviewfocusedonthe followingtopics: ○ Securechannels.Secureend-to-endcommunicationisimplementedusing theNoiseframework,awell-knownframeworkfordesigningsecurechannel protocols.OckamSecureChannelsarebasedonthe XX patternoftheNoise protocol.Wereviewedtheuseofsecurechannelsin Ockam,focusingon whethertheNoiseprotocolisproperlyinstantiatedandusedeffectivelyto provideend-to-endsecurity.Wepaidspecialattentiontotheeffectof unreliabledeliveryonthesecurityofsecurechannels.Wealsoinvestigated authenticationguaranteesofthesecurechannelswhenusedina post-specifiedsettingwherenodesmayestablishsecurechannelswithother nodeswhoseidentitiesarenotknowninadvance. ○ Routingandtransportmechanisms.WithinOckam’sinfrastructure,routing andtransportmechanismsdelivermessageswithnoreliabilityguarantees outofthebox,unlikeprotocolssuchasTCPorQUIC.Additionalmechanisms areusedinthesystemtoprovidesomeformsofreliability,buttheseare currentlynotdocumentedinthedesign.Weinvestigatedtheroutingand transportmechanismsusedinOckam,focusingontheirimpactonthe securityofsecurechannels.Moreover,weexaminedpotentialconcerns leadingtodenialofserviceorwasteofresources. ○ Identitiesandcredentials.IntheOckamsystem,eachparticipatingentity hasoneormoreOckamIdentities.Thesearecryptographicallyverifiable digitalidentitiesthatallowotherentitiestoverifytheauthenticityofdigital signaturekeysusedtoauthenticateusers.Tobootstraptrustinthese identities,theOckamsystemsupportstheuseofcredentials,whichare signedattestationsofidentitiesprovidedbyanissuer.Weinvestigatedthe cryptographicprimitivesusedtorealizetheseconcepts,aswellasthe applicabilityofknowncryptographicattacks. ○ Accesscontrolsandpolicies.TheOckamsystemreliesonvarious cryptographictechniquestoguaranteeauthenticityandconfidentiality.Each particularusecaseintheOckamsystemcanuseaccesscontrolsandaccess controlpoliciestoensurethatcertainchecksareperformed,suchasthe verificationofcredentialsfromaspecifiedissuer.Weinvestigatedwhether hasheye 10OckamDesignReview PUBLIC

parsingissuescouldcircumventthesesecuritycontrolsandwhetherthey werecorrectlyappliedinthetwoin-scopeusecases. ○ TCPPortalsandKafkaPortalsusecases.Theseusecasesfocuson enablingend-to-endencryptedcommunicationbetweenendpointsoverany networktopology.TheTCPPortalsusecasefocusesonenablingapplications tocommunicatewitheachotherasiftheyweremakingalocalTCP

connection,whereastheKafkaPortalsusecaseachievesthesamefor applicationsthatconsidertheyaretalkingtoKafkaserversviaalocal connection.Weinvestigatedtheuseofthird-partyinfrastructure(suchas GitHub,AWS,OAuth,etc.),thecorrectuseofdifferentaspectssupportedby theoverallOckamsystem,andwhetheranyissuesinspecificsystem componentscouldbeusedtocompromiseconfidentialityorintegrityofuser data. • Q&Asessionswiththedesigners.Incaseswherethedocumentationdidnot providesufficientinformationtodeterminetheapplicabilityorimpactofpotential attacks,weaskedquestionsofthedesigners,whoprovidedanswers. • Formalmodelingofprotocolaspects.WeusedVerifpalandCryptoVeriftomodel thefollowingaspects: ○ ModelingwithVerifpal.WeusedVerifpaltomodelthecryptographiccore ofsecurechannels.Wefocusedonconfidentialityandauthentication guarantees.WealsomodeledidentitiesinOckamtounderstandwhetherthe intendedauthenticationguaranteesareprovided. ○ ModelingwithCryptoVerif.WemodeledidentitiesinCryptoVerif,which allowedustofurtherunderstandtheexactsecuritypropertiesneededto ensurethatthedesignprovidedsufficientguaranteesforeachintendeduse case. CoverageLimitations Becauseofthetime-boxednatureoftestingwork,itiscommontoencountercoverage limitations.Wedidnotformallymodelallaspectsofthesystem.Specifically,thefollowing maywarrantfurthermodeling(althoughallwereconsideredaspartofthemanualreview): • Identitybindingguaranteesofsecurechannels.Thesecurechannelprotocol allowspost-specifiedpeers,wherecommunicatingpartiesmaylearnaboutthe peer'sidentityuponcompletionoftheprotocol.Insuchscenarios,theprotocol mustproperlybindidentitiestothederivedsessionkey.Inparticular,theprotocol mustprotectagainstidentitymisbindingattacks,alsoknownasunknownkey-share attacks.OckambuildsuponNoise’spublickeybindingtobindidentitiestosession hasheye 11OckamDesignReview PUBLIC

keys.Duetotimelimitations,wedidnotformallymodelwhethersuchbindingwas donesecurely. • Completemodelingofidentitiesandcredentials.Inourformalmodelingwork, wefocusedonmodelingthecryptographiccoremechanismsunderpinning identities,changehistories,andcredentials.Indoingso,weabstractedsome elements,focusingonlyoncryptographicallyrelevantaspects.Nevertheless,more completemodelingwillprovidevaluableinsightsintothesecurityofthedifferent componentsusedinOckam. • Post-compromisesecurityofvariouskeys.Wedidnotmodelany post-compromisesecurityofidentitykeys,purposekeys,orsessionkeys.Itisworth investigatingwhatguaranteescanstillbeprovidedwhenspecifickeysare compromised. hasheye 12OckamDesignReview PUBLIC

AutomatedTesting hasheyeusesautomatedtechniquestoextensivelytestthesecuritypropertiesof softwareandprotocols.Weusebothopen-sourcestaticanalysisandfuzzingutilities,along withtoolsdevelopedinhouse,toperformautomatedtestingofsourcecodeandcompiled software.Wefurtheruseopen-sourceformalmodelingtoolstoperformautomatedtesting ofcryptographicprotocolsandsystemdesigns. TestHarnessConfiguration Weusedthefollowingtoolsintheautomatedtestingphaseofthisproject: ToolDescriptionPolicy VerifpalAnopen-sourcesymbolicprotocolanalyzerthatfocuseson easeofuse,allowingrapidmodelingandverificationof cryptographicprotocols AppendixC.1 CryptoVerifAnopen-sourcecomputationalprotocolverifierandproof assistantusedtocreatemodelsthatstateprecise assumptionsontheprimitivesusedintheprotocolandthe desiredsecuritypropertiestobeproven AppendixC.2 AreasofFocus Ourautomatedtestingandverificationworkfocusedonthefollowingsystemproperties: • AuthenticityofOckamIdentities • Confidentialityandauthenticityofdataexchangedduringthesecurechannel handshake Theresultsofthisfocusedtestingaredetailedbelow. AuthenticityofOckamIdentities.WeusedVerifpaltocreateamodelofOckamIdentities inascenariowhereoneusercommunicatestwo Change blockstoanotheruserthathas accesstothefirstuser’sidentifier.WealsousedCryptoVeriftocreateamodelofchange generationinascenariowhereanadversarycanobtainsignaturesonarbitrarymessages undertheprimaryidentitykeys. PropertyToolResult Authenticityoffirst Change blockbasedonOckamIdentifierVerifpalPassed hasheye 13OckamDesignReview PUBLIC

PropertyToolResult Authenticityofsecond Change blockbasedonOckam Identifier VerifpalPassed Existentialunforgeabilityof Change blocksCryptoVerifPassed Strongunforgeabilityof Change blocksCryptoVerifTOB-OCK-6 Thelinkedfindingaboveisinformationalseverityandnotcurrentlyexploitable;pleasesee thefindingitselfforfurtherdetails. OckamSecureChannels.WeusedVerifpaltomodelthecryptographiccoreofOckam

SecureChannels,includingbutnotlimitedtothehandshake. PropertyToolResult SecrecyofsessionkeysVerifpalPassed ConfidentialityandauthenticityoftransportedmessagesVerifpalPassed ThedesignprinciplesoftheNoiseframeworkenablethe XX patterntoprovidefurther securityguarantees.Forinstance,the XX patternisdesignedtoberesistanttokey compromiseimpersonation(KCI)attacks.KCIresistanceguaranteesthatevenwhena user’slong-termsecretkeyiscompromised,theadversaryshouldnotbeableto impersonateotheruserstowardstheuserwhosekeywascompromised.Thisresulthas beenfurtherconfirmedbyDowling,Rösler,andSchwenkintheiranalysisoftheNoise framework.Therefore,wedidnotverifytheseguaranteesusingourformalmodel. hasheye 14OckamDesignReview PUBLIC

SummaryofFindings Thetablebelowsummarizesthefindingsofthereview,includingtypeandseveritydetails. IDTitleTypeSeverity 1ThesystemisvulnerabletoSNDLattacksby quantumcomputers CryptographyMedium 2SerializedVersionedDatastruct’sdatais ambiguous CryptographyInformational 3TruncatingChangeHistoryhashto160bits introducesriskofcollisions CryptographyInformational 4Themeaningsoftheprimarykeyfieldscreated_at andexpires_atareundocumented CryptographyMedium 5InsufficientthreatmodeldocumentationCryptographyMedium 6Thesupportedsignatureschemeshavedifferent securityproperties CryptographyInformational hasheye 15OckamDesignReview PUBLIC

DetailedFindings 1.ThesystemisvulnerabletoSNDLattacksbyquantumcomputers Severity:MediumDifficulty:High Type:CryptographyFindingID:TOB-OCK-1 Target:KeysandVaultssection Description TheOckamsystemusestheNoise XX handshakepatternwithECDHbasedonX25519, whichcouldbebrokenusingalarge-scalequantumcomputer.Currently,noquantum computercapableofdoingthisexists.AlthoughOckamcouldbequicklyupdatedtoresist quantumattacks,currenthandshakeswouldstillbevulnerableto"storenow,decrypt later" (SNDL)attacksusingaquantumcomputer,asdescribedintheexploitscenario below.Thedesignshouldaddresswhetherthisisconsideredaspartofthethreatmodel. ExploitScenario Anattackercapturesandstoresthefulltranscriptofahandshakeandthesubsequent encryptedcommunications.Oncealarge-scalequantumcomputerbecomesavailable,the attackerrecoversallECDHprivatekeysusingthequantumcomputer.Theyusethisto obtainthederivedkeysfromthetranscriptanddecryptthecommunications. Recommendations Shortterm,documentwhetherSNDLattacksusingaquantumcomputerareapplicableto thethreatmodelfordifferentusecasessothatuserscanconsiderthisinthecontextof theirownriskmanagement.Iftheattacksareapplicable,investigatethefeasibilityof incorporatingpost-quantumsecurealternativesintothesystem. BecausethegoalistopreventSNDLattacks,itisnotnecessarytoupgradeall cryptographicprimitivestobesecureagainstaquantumcomputer.However,atleastone ofthecontributionstothekeyderivationmustcomefromaPQCKEM(e.g.,Signal’s PQXDH).AsdescribedinthePQNoisepaper,itisstraightforwardtoupdatetheNoise ee patternusingaPQCKEMtoachievethesameroundcomplexity.Insteadofreplacingthe classicalDH,weproposeaddingaPQCKEMtoachieveahybridsolution(e.g.,theIETFdraft onhybridkeyexchangeinTLS). Longterm,ifattacksusingaquantumcomputerarepartofthethreatmodelforOckam usecases,migrateboththeKeyexchangesanddigitalsignaturestoahybridsolution hasheye 16OckamDesignReview PUBLIC

incorporatingbothclassicalandpost-quantumresistantprimitives(e.g.,usingPQNoise withsuitablehybridKEMs). hasheye 17OckamDesignReview PUBLIC

2.SerializedVersionedDatastruct’sdataisambiguous Severity:InformationalDifficulty:High Type:CryptographyFindingID:TOB-OCK-2 Target:IdentitiesandCredentialssection Description Identityprivatekeysareusedtosignboth Change blocks,whichareusedtorotatethe identitykeys,and PurposeKeyAttestation blocks,whichareusedtoattesttopurpose keys.Beforebeingsigned,thedatainsidetheseblocksisserializedusingtheConcise BinaryObjectRepresentation(CBOR)dataformatandincludedina VersionedData struct. However,thisserializationwillnotadddifferentlabelsforthesedifferentdatatypesby itself,whichmeansthatthe data fieldofa VersionedData instancedoesnotindicate whichtypeitcontains(i.e., ChangeData or PurposeKeyAttestationData).Therefore,it ispossibletoprovideasigned VersionedData instancecontaininga PurposeKeyAttestationData blockwherethereceiverexpectsittocontaina ChangeData block.Thereceiverwillpotentiallyacceptthesignatureasvalidaslongasitis createdusingtheexpectedidentitykey. Whetherthisconfusioncanbeexploiteddependsonimplementationdetailsthatarenotin scopeforthisdesignreview.Specifically,itdependsonwhathappenswhena PurposeKeyAttestationData typeisdeserializedasa ChangeData type.Currently,it seemslikelythatthiswillfailduetodifferencesinthestructureofthetypes,butitisnot

possibletodeterminethefullbehaviorfromthedesigndescriptionalone.Evenifthe currentimplementationrejectsaserialized PurposeKeyAttestationData instanceas invalidwhenitattemptstodeserializethebytevectorasa ChangeData structure,afuture versionoftheprotocolmayacceptitasvalidifeitherofthetwotypeschange. ExploitScenario Anodecreatesandatteststovariouspurposekeystooutsourcethehandlingofthe purposetootherinstances,whichdonothaveaccesstotheidentitykey.Anattacker compromisesoneoftheinstancesandobtainsthe PurposeKeyAttestation block containingasigned VersionedData instance. Theattackercommunicatesanew Change toanothernode,whileprovidingthesigned VersionedData fromthe PurposeKeyAttestation block.The previous_signature fieldinsidethe Change blockissettothe signature fromthe PurposeKeyAttestation block(i.e.,thesignatureonthe VersionedData instancecontainingthe hasheye 18OckamDesignReview PUBLIC

PurposeKeyAttestationData).Theothernodewillacceptthe previous_signature on the VersionedData becauseitisasignatureunderthesameidentityprimarykey. Forthisexploittobeuseful,the data fieldofthe VersionedData instance,whichisa serialized PurposeKeyAttestationData type,mustdeserializetoa ChangeData type withapublickeyforwhichtheattackerknowsthecorrespondingprivatekey.Otherwise, theattackerwillbeunabletoprovideaproper signature forthe Change block. Recommendations Shortterm,specifyinthedesignthatthe data fieldinsidethe VersionedData structtobe signedusingidentityprimarykeysmustbeunambiguous.Forexample,thisfieldcouldbea single enum typethatcontainsthe PurposeKeyAttestationData and ChangeData types withdifferentCBORlabels.Alternatively,addanadditionallabeltothe VersionedData structtoindicatewhichdatatypeitcontains. Longterm,ensurethatalldifferenttypesofprivatekeysusedtocreatedigitalsignatures signonlyasingleunambiguoustype(withdifferentlabelsforeachofthepossiblesubtypes orcontainedtypes). hasheye 19OckamDesignReview PUBLIC

3.TruncatingChangeHistoryhashto160bitsintroducesriskofcollisions Severity:InformationalDifficulty:High Type:CryptographyFindingID:TOB-OCK-3 Target:IdentitiesandCredentialssection Description The Identifier and ChangeHash typesaredefinedasthefirst160bitsofaSHA-256hash ofthe Change recordinsideofa ChangeHistory chain.AcollisionattackagainstSHA-256 truncatedto160bitsispossiblewithatimecostofapproximately2 80 queriesusing standardcollision-findingtechniquesbasedonPollard’sRhomethod.Thesefiguresareon theupperendoffeasibilitybutstillpossibletoexploit. ExploitScenario Mallory,auserwithanexistingOckamIdentitywantstocausedisagreementabouther currentprimarypublickeybetweendifferentnodes,whoarerelyingonthelatest ChangeHash toidentifytheuser’sprimarypublickey. SheusesPollard’sRhomethodtofindtwo ChangeData recordswiththesame ChangeHash ,whichallowshertocreatetwodistinct ChangeHistory chainswherethe finalentriesdisagreeontheprimarypublickeybutstillhavethesame ChangeHash . Computing2 80 SHA-256hashesiswithinthereachofbotnetstoday.Forcomparison,the Bitcoinminingnetworkcomputesabout2 68 SHA-256hashespersecond(orabout2 92 per year).Theequivalentamountofcomputationalresourcescancrossthe2 80 thresholdina littleover1hour. Onceacollisionisfound,Mallorythenselectivelysharesdifferent Change recordsto differentpartitionsofthenetwork. Recommendations Shortterm,increasethelengthofthetruncatedSHA-256hashfrom160bits(20bytes)to atleast192bits(24bytes)forthe ChangeHash .Withthischange,thecostoffindinga collisionbecomes2 96 ratherthan2 80 ,whichmultipliesthetimerequiredtofindacollision byafactorof65,536.Consequently,thecollisionattackisnolongerpractical. Longterm,wheneverreducingthesecuritymarginofacryptographicprimitive(e.g., truncatinghashesinthiscase),documentwhythisisdoneandwhytheimpactonsecurity isconsideredacceptable. hasheye 20OckamDesignReview PUBLIC

4.Themeaningsoftheprimarykeyfieldscreated_atandexpires_atare undocumented Severity:MediumDifficulty:High Type:CryptographyFindingID:TOB-OCK-4 Target:IdentitiesandCredentialssection Description Themeaningsofthe created_at and expires_at fieldsonthe ChangeData typearenot sufficientlyexplainedbythedesigndocumentation.Thismayleaduserstomake security-criticaldecisionsbasedonaflawedinterpretationofthesefields. The ChangeData typeisusedtoupdateauser’sprimarykey.Thetype’stwofields, created_at and expires_at ,definethelifetimeofthenewkey.TheIdentitiesand Credentialssectionmentionsthatthe expires_at timestampindicates“whenthe primary_public_key includedinthechangeshouldstopbeingreliedonastheprimary publickeyofthisidentity.” However,thedocumentationdoesnotspecifywhathappensifauserallowstheirprimary keytoexpirewithoutsigningandbroadcastinganewchange.Intuitively,itiseasyto assumethatnodesacceptchangesonlyfromlivekeys,soanexpiredkeycannolonger signnewchanges.However,thisisnotdescribedinthedocumentationandthereis currentlynocheckinthecodeperformingchangehistoryvalidationtoensurethis behavior. Themeaningofthe

created_at fieldisalsonotsufficientlyexplained.Itiscurrently
unclearhowthisfieldshouldbevalidatedoractedonbyothernodes.Infact,the
implementationexplicitlyallowschangeswhere created_at isgreaterthan expires_at . Thismeansthatthe
created_at fieldcannotbereliedontodefineanoveralllifetimeor validityperiodforthekey.
ExploitScenario AliceusesOckamtosetupanetworkofnodes.Oneofthenodesistakenoffline,and
eventuallytheprimarykeyusedforthenodeexpires.Sincethekeyhasexpired,Alice
believesthatitisnolongersensitiveanddoesnottakeproperprecautionstoeitherprotect ordeletethekey.
Mallory,amalicioususer,gainsaccesstothenodeandobtainstheexpiredkey.Shecan
nowcreateanewchangebasedontheexpiredkey.Shebroadcaststheupdatedchange
historytoothernodesinthenetwork.Sincenodesdonotcheckthe expires_at field hasheye
21OckamDesignReview PUBLIC

whenthechangehistoryisvalidated,thenewchangeisacceptedasvalidbyothernodes,
allowingMallorytogainaccesstothenetwork. Recommendations Shortterm,documenttheexpectedmeaningsofthe
created_at and expires_at fields,
andspecifyhowthesefieldsshouldbevalidated.Ensurethatchangessignedbyexpired
keysarerejectedbyallnodes.Alternatively,ifthelifetimeismeanttobeenforcedonlyfor
purposekeyattestations,documentthisrestrictionandrenamethetwofields(e.g.,to
attestations_not_valid_before and attestations_not_valid_after)tomake thisclear.
Additionally,clearlyspecifythefulllifecycleofsecretsandcredentials,includingany
applicablerevocationorexpirationmechanisms.
Longterm,ensurethatthedocumentationalwaysreflectsthepropermeaningofeach
valuespecifiedbytheprotocol.Inparticular,ifvalueshaveunintuitiveorsurprising
meanings,theyshouldalwaysbedocumented. hasheye 22OckamDesignReview PUBLIC

5.Insucientthreatmodeldocumentation Severity:MediumDifficulty:NotApplicable
Type:CryptographyFindingID:TOB-OCK-5 Target:Allsections Description
ThethreatmodelforOckam’sprotocolsisnotspecifiedinthedocumentation.Thereisno
descriptionofthedifferentactorsinthesystem,theassetstheyvalue,theotheractors
theytrust,andthesecuritycontrolsforachievingsecuritygoals.Fromthethreatmodel,it
shouldbeclearwhataspectsoftheassetsareimportant,includingbutnotlimitedto
confidentiality,integrityorauthenticity,andavailability.
Thedesignreviewincludedweeklydiscussionsbetweenauditorsanddevelopers,who
providedallrelevantthreatmodelinginformationforthetwoin-scopeusecases.However,
intheabsenceofadocumentedgeneralthreatmodelfortheprotocol,usersand
developersmustmakeassumptionsaboutthesecuritygoalsofeachcomponentandhow
thesegoalsaremetintheimplementation.Ifanyoftheseassumptionsarefalse,thiscould
leadtosurprisingbehaviorandpotentiallyrealsecurityissueswhenusersdeploythe protocol.
Thereisnospecificexploitscenarioforthisfinding,sothedifficultyratingisnotapplicable.
Recommendations Shortterm,addaninformalthreatmodeltoeachusecaseandsectionofthe
documentationtoensurenogapsexist.
Longterm,developaformalthreatmodelthatappliestoOckam’sprotocols.Explicitlystate
anyassumptionsthatmustbetrueforthe protocoltobesecure.Definedifferenttypesof
threatactorsandspecifyhowtheprotocoldealswiththem.
Onceageneralthreatmodelisinplace,eachusecaseandsectionoftheprotocolneedsa
threatmodelsectionthatdescribesonlythewaysinwhichtheydeviatefromthegeneral
modelfortheoverallprotocol. hasheye 23OckamDesignReview PUBLIC

6.Thesupportedsignatureschemeshavedierentsecurityproperties
Severity:InformationalDifficulty:NotApplicable Type:CryptographyFindingID:TOB-OCK-6
Target:Allsections Description
Ockam’ssupportedsignatureschemes,EdDSAandECDSA,offerdifferentsecurity
guaranteesbutareusedinterchangeably.Althoughthisissueisnotcurrentlyexploitable,if
Ockammodifiesthesysteminthefuture,itcouldbecomenecessarytorelyonadditional securityguarantees.
ThesignedchangehistoryinOckamallowsanidentitytorotateitsprimarykeyandattest
tothevalidityofthischange.Eachrotationisassociatedwitha Change datastructurethat
ishashedandthensignedbythecurrentprimarykeyandtheprevious primarykey(ifit
exists).Thedocumentationspecifiestwosigningalgorithms: EdDSACurve25519Signature and
ECDSASHA256CurveP256Signature .Theformeris
thepreferredalgorithm,asthelatteralgorithmissupportedonlyduetothelackofsupport
forEdDSAincloudhardwaresecuritymodules(HSMs).Aformalmodelingofthesecurity
propertiesofthesignatureschemewithCryptoVerifrevealsadiscrepancybetweenthe
securityguaranteesofferedbythetwoschemes.
Intermsofsecurityguarantees,EdDSAandECDSAareequivalentonlyinthattheyareboth
believedtoguaranteeExistentialUnforgeabilityunderChosen-MessageAttacks(EUF-CMA).
Thismeansthatanattackerwhohasseveralvalidsignaturesonvariousmessageswillbe

unabletocreateavalidsignatureforanewmessage.
However,EdDSA,asinstantiated,providesadditionalguaranteesthatECDSAdoesnot.In particular,EdDSAprovidesStrongUnforgeabilityunderChosen-MessageAttacks (SUF-CMA),soanattackerwhohasanumberofvalidsignaturesonvariousmessageswill beunabletocreateavalidnewmessage-signaturepair.
ECDSAdoesnotprovidethisguaranteebecauseforanyvalidECDSAsignature(r , s),the signature(r , -s)isalsovalid.Thismeansthatanattackercouldtakea Change blockwith associatedECDSAsignaturesandsharethesame Change blockwithmodified(butvalid) signatures.WhenmodelingchangehistorieswithCryptoVerif,onlythestrong unforgeabilityofthefirstchangecanbeproven.
Theissuedescribedabovedoesnotposeadirectthreatinthecurrentdeploymentof Ockaminthecontextofthetwoin-scopeusecases,whichdonotrelyonstrong hasheye 24OckamDesignReview PUBLIC

unforgeability.However,fromadesignperspective,itisdesirabletohaveaclear understandingofwhatsecurityguaranteesareexpectedfromdifferentcomponents, irrespectiveoftheirconcreteinstantiations.Furthermore,anyfutureextensiontothe protocolmightinvolvebeyondunforgeabilityguaranteeslikeexclusiveownership(which meansthatanyvalidsignaturecanbecreatedonlyfromtheprivatekeycorrespondingto thepublickey).Fortunately,thecurrentsigningmechanisminOckamisclosetotheBUFF construction,whichprovidesbeyondunforgeabilitysecurity.
Thereisnospecificexploitscenarioforthisfinding,sothedifficultyratingisnotapplicable.
Recommendations Shortterm,consideraddingabriefdescriptionofthesecuritypropertiesofthesigned changehistoriesandothersigneddatastructures.Thendocumenthowtheinstantiations ofdifferentcomponentsprovidetheexpectedsecurityguarantees.Considerallusecases ofsignaturesinthesystemandwhetheranybeyondunforgeabilityguaranteesmightbe expected.IfSUF-CMAsecurityisdesired,ECDSAcanbemodifiedbyrestrictingthe s componentofthesignaturetotheupperorlowerhalfofitsrange.
Longterm,specifyallthecryptographicassumptionseachsystemcomponentisexpected tomeetfortheprotocoltobesecure.Documenthoweachinstantiationofaspecific primitivemeetstherequiredassumption. hasheye 25OckamDesignReview PUBLIC

A.VulnerabilityCategories
Thefollowingtablesdescribethevulnerabilitycategories,severitylevels,anddifficulty levelsusedinthisdocument. VulnerabilityCategories CategoryDescription
AccessControlsInsufficientauthorizationorassessmentofrights
AuditingandLoggingInsufficientauditingofactionsorloggingofproblems
AuthenticationImproperidentificationofusers
ConfigurationMisconfiguredservers,devices,orsoftwarecomponents
CryptographyAbreachofsystemconfidentialityorintegrity DataExposureExposureofsensitiveinformation
DataValidationImproperrelianceonthestructureorvaluesofdata
DenialofServiceAsystemfailurewithanavailabilityimpact
ErrorReportingInsecureorinsufficientreportingoferrorconditions
PatchingUseofanoutdatedsoftwarepackageorlibrary
SessionManagementImproperidentificationofauthenticatedusers
TestingInsufficienttestmethodologyortestcoverage TimingRaceconditionsorotherorder-of-operationsflaws UndefinedBehaviorUndefinedbehaviortriggeredwithinthesystem hasheye 26OckamDesignReview PUBLIC

SeverityLevels SeverityDescription
InformationalTheissuedoesnotposeanimmediateriskbutisrelevanttosecuritybest practices.
UndeterminedTheextentoftheriskwasnotdeterminedduringthisengagement.
LowTheriskissmallorisnotonetheclienthasindicatedisimportant.
MediumUserinformationisatrisk;exploitationcouldposereputational,legal,or moderatefinancialrisks.
HighTheflawcouldaffectnumeroususersandhaveseriousreputational,legal, orfinancialimplications.
DifficultyLevels DifficultyDescription
NotApplicableWedidnotidentifyaspecificexploitscenarioforthisfinding.
UndeterminedThedifficultyofexploitationwasnotdeterminedduringthisengagement.
LowTheflawiswellknown;publictoolsforitsexploitationexistorcanbe scripted.
MediumAnattackermustwriteanexploitorwillneedin-depthknowledgeofthe system.
HighAnattackermusthaveprivilegedaccesstothesystem,mayneedtoknow complextechnicaldetails,ormustdiscoverotherweaknessestoexploitthis issue. hasheye 27OckamDesignReview PUBLIC

B.DesignQualityFindings Weidentifiedthefollowingdesignqualityissuesbyreadingthedesigndocumentation.
Thesearegenerallyminorinconsistenciesandtyposthatdonotcorrespondtosecurity vulnerabilities.However,resolvingthemisrecommendedincasepartsofthedesign

documentationarereusedincustomer-facingdocumentation: • ThedescriptionofthederivationofOckamIdentitiesisnotconsistentwiththe implementation.Itstatesthattheidentifierisahashofa Change block,which includesasignature.However,theidentifierisahashofonlytheincluded ChangeData type. • TheimageinthesectiononPurposeKeyAttestationsismissingapurposeinthe PurposeKeyAttestationData type(cf.thenextimageintheCredentialssection). • TheimageontheNoisehandshakeinthesectiononAuthenticatedKey Establishmentdoesnotdescribetheconstructionofthepayloadscontainingthe IdentityAndCredentials typeofboththeresponderandinitiator. • Thedetailsofthe Rekey operationarenotspecifiedinthedesigndocumentation. Thespecificationmentionsthatarekeyisdoneviaanencryptionoperation. However,thedetailsofwhatisbeingencryptedaremissing,andthespecification linksonlytotheNoisespecificationforrekeying.Forcompleteness,includea descriptionofrekeyingintheOckamdocumentationaswell.Thisisalsomore consistentwiththedescriptionofthehandshake,whichincludesmoredetailsfrom theNoisespecification. • Theslidingwindowmechanismisusedtomitigatereplayattacks.However,the exactdetailsofhowthisworksarenotincludedinthedescriptionofthesliding windowmechanismbutarecoveredonlyimplicitlybytheexamples. • ThetitlefortheTrustContextssectionismisspelled. • IntheTCPPortalsusecasedescription,therearethreeoccurrencesofthetypo “idenity.” • IntheKafkaPortalsusecasedescriptiononcreatingaKafkaproducer,thetext statesthatthecommandtocreateaKafkaProducer“createsasidecartoaKafka Consumer.”Itfurtherstatesthatcertaincomponentsandresponses“workexactly thesameasintheproducersidecarandtheconsumersidecarabove,”butitshould referonlyto“theconsumersidecarabove.” hasheye 28OckamDesignReview PUBLIC

C.FormalModeling Duringtheengagement,weusedthesymbolicmodelcheckerVerifpaltoformallymodel differentaspectsoftheprotocol.Thisworkandtheresultsobtainedaredescribedinthis section.

C.1VerifpalModeling Verifpalisasymbolicmodelchecker,muchlikeTamarinandProVerif.Itsmainadvantage overothertoolsisthatitfocusesonusabilityfirst,whichallowstheusertoquicklymodel newprotocolsandupdatestoexistingprotocols. SymbolicmodelcheckerslikeVerifpalmodelcryptographicprimitivesasblackboxes. Protocolparticipantsandanattackercanperformcomputationsusingthedefined cryptographicprimitives,andalgebraicpropertiesoftheseprimitivesaremodeledusing equations.Theattackerisallowedtoruntheprotocolanunboundednumberoftimesto observe(inthecaseofapassiveattacker)andmodify(inthecaseofanactiveattacker) messagesinflight. (ThisissometimesreferredtoastheˇDolev-Yaomodel.)Inthismodelit ispossibletoproveequivalencebetweenterms(e.g.,asharedsecretcomputedindividually bytwoprotocolparticipants),aswellasmodelcommonsecuritypropertieslike confidentialityandauthenticityofmessagesbetweenparticipants.Itisalsopossibleto analyzemorecomplexsecuritypropertieslikeforwardsecrecy,post-compromisesecurity, andsecurityagainstKCIattacks. However,therearelimitstowhatcanbeexpressedinthesymbolicmodel.Since cryptographicprimitivesaremodeledinanidealizedway,thereisnowaytoanalyze attackeradvantageorasymptoticsecurityproperties.(Thesetypesofpropertiesaremore naturallyexpressedinthecomputationalmodelusingtoolslikeCryptoVerif.)

ModelingOckamIdentitiesUsingVerifpal WeusedVerifpaltomodelOckamIdentitiesandchangehistoryvalidation.Thegoalofthis exercisewastoformallyprovethatanattackercouldnottamperwithupdatestothe changehistorywithoutitbeingdiscovered.Thefollowingmodelimplementschangehistory validationundertheassumptionthatidentitiesaredistributedoutofband.Forsimplicity, wedisregardprotocolversioning,purposekeyrevocation,andprimarykeylifetimes becausethesefieldsareirrelevantforthesecuritypropertywearetryingtomodel. attacker[active] //Alicecreatesherfirstprimarykeypair. principalAlice[knowsprivatesecret_key_0 //Generate`ChangeData`0. public_key_0=G^secret_key_0 hasheye 29OckamDesignReview PUBLIC

change_data_0=CONCAT(nil,public_key_0) //Computeidentityasthehashofthefirst`ChangeData`andsignthefirsthash. change_hash_0=HASH(change_data_0) signature_0=SIGN(secret_key_0,change_hash_0)] //Aliceupdatesherprimarykeypair. principalAlice[knowsprivatesecret_key_1 //Generate`ChangeData`1andsignitshash. public_key_1=G^secret_key_1 change_data_1=CONCAT(change_hash_0,public_key_1) change_hash_1=HASH(change_data_1) signature_1=SIGN(secret_key_1,change_hash_1) previous_signature_1=SIGN(secret_key_0,change_hash_1)] //Alicenowsendsher`ChangeHistory`toBob. //Alice'sidentity(`change_hash_0`)issharedoutofband. Alice→Bob:[change_hash_0],change_data_0,signature_0//nil Alice->Bob:change_hash_1,change_data_1,signature_1,previous_signature_1

FigureC.1.1:Alicecreatestwochanges,whicharethensenttoBob.
//BobverifiesAlice'sfirst`Change`againstheridentity. principalBob[
//Verifythefirst`ChangeHash`againstAlice'sknownidentity.
valid_0=ASSERT(HASH(change_data_0),change_hash_0)?
//Verifythe`ChangeHash`signatureagainstthecurrentkey.
alice_previous_change_0,alice_public_key_0=SPLIT(change_data_0)
_=ASSERT(alice_previous_change_0,nil)? _=SIGNVERIF(alice_public_key_0,change_hash_0,signature_0)?]
//Dummymessagesignalingthatthe`ChangeData`receivedbyBobwasaccepted. Bob→Alice:[valid_0]
//BobverifiesAlice'ssecond`Change`againstherfirstprimarypublickey. principalBob[
//Verify`ChangeHash`signatureagainstthepreviouskey.
valid_1=SIGNVERIF(alice_public_key_0,change_hash_1,previous_signature_1)?
_=ASSERT(HASH(change_data_1),change_hash_1)? //Verify`ChangeHash`signatureagainstthecurrentkey.
alice_previous_change_1,alice_public_key_1=SPLIT(change_data_1)
_=ASSERT(alice_previous_change_1,change_hash_0)?
_=SIGNVERIF(alice_public_key_1,change_hash_1,signature_1)?]
//Dummymessagesignalingthatthe`ChangeData`receivedbyBobwasaccepted. hasheye 30OckamDesignReview PUBLIC

Bob→Alice:[valid_1] //Alicehastouseallmessagesreceivedinorderforthemodeltocompile. principalAlice[
_=HASH(valid_0) _=HASH(valid_1)]
FigureC.1.2:BobvalidatesthetwochangesagainstAlice’sidentity,whichissentoutofband. queries[
//TheattackercannothavetamperedwithAlice'sfirst`Change`underthe
//assumptionthatBobacceptsthefirst`ChangeData`asvalid. authentication?Alice→Bob:change_data_0[
precondition[Bob→Alice:valid_0]] //TheattackercannothavetamperedwithAlice'ssecond`Change`underthe
//assumptionthatBobacceptsthesecond`ChangeData`asvalid. authentication?Alice→Bob:change_data_1[
precondition[Bob→Alice:valid_1]]]
FigureC.1.3:Themodelcheckswhethertheattackercouldhavetamperedwitheitherofthetwo ChangeData
messagesundertheassumptionthatBobacceptedthemessageasvalid.
WemodeledtheprotocolusinganactiveattackerundertheassumptionthatAlice’s
identityissharedoutofbandwithBob.InVerifpal,thiscanbeachievedusingguarded
messages(denotedusingsquarebrackets).Sinceeachchangeisvalidatedinmultiple
steps,wehadtoindicatetotheproverthatBobmaycomputeonuntrusteddataaspartof
thevalidationprocess.Forthisreason,wemodeledauthenticationundertheadditional
assumptionthatBobsuccessfullyvalidatesthecorrespondingchange,whichisindicatedby
sendingadummymessage(denoted valid_0 and valid_1 above)toAlice.
Runningtheproveronthismodeldidnotidentifyanyattacksontheprotocol.Thisoffersa
highlevelofconfidencethatthedesignissecureagainstactiveattacks.
Whennotincludingtheadditionalassumptionandthedummymessages,Verifpal
describesthefollowinghypotheticalattack: 1.Anattackerrecordsthefirst Change anditsself-signature.
2.Theattackerreplacesthesecond Change withthefirst Change ,settingboth
signaturestobetherecordedself-signature. 3.NowBobwillverifythesignatureonthesecond Change
usingthepublickeyofthe first Change ,whichwillbecorrectsinceitistheself-signature. hasheye
31OckamDesignReview PUBLIC

4.Next,Bobwill SPLIT the Change data,whichVerifpalconsidersacomputationon attacker-
provided(andhencenon-authentic)data.
ItisclearthatthisattackdoesnottranslatetoOckamIdentitiesbecausethenextstepof
theverificationwouldcheckthe previous_change_hash inthe Change dataagainstthe hashofthefirst Change
,whichwillfailinthiscase.However,theattackdoeshighlighta smallissueinthedesign,whichisthatthe
previous_signature and signature fieldsof a Change
arenotdomainseparatedandanattackercanexchangeorcopythem.Itisworth
consideringfixingthisissue,althoughitiscurrentlynotexploitable.
ToreanalyzethemodelusingVerifpal,savethemodeldefinedinfiguresC.1.1-C.1.3above asasinglefilecalled
ockam-identities.vp ,andthenrunVerifpalonthemodelusingthe followingcommand: verifpalverifyockam-
identities.vp C.2CryptoVerifModeling
CryptoVerifisacryptographicproofassistantandprotocolverifierthatoperatesinthe
computationmodel.Unlikesymbolicverifiers,inCryptoVerif,messagesarebitstrings,
cryptographicprimitivesarefunctionsmanipulatingbitstrings,andtheadversaryis
modeledasaprobabilisticpolynomial-timeTuringmachine.Cryptographicproofsarethen carriedoutinthegame-
playingframeworkpioneeredbyBellareandRogaway(among
others).Concretely,thesecurityofaprotocolisanalyzedthroughagameplayedbetweena
challengerandanadversary.Theadversaryisgivenaccesstoseveraloraclesthatcapture
theadversary’scapabilities(e.g.,theadversarymayobtainsignaturesoverarbitrary
messages).Finally,awinningconditiondefineswhatactiontheadversarymustperformfor
agivensecuritypropertytobebroken(e.g.,theadversarymustproduceasignature forgeryforanewmessage).

CryptoVerifprovessecurityusingsequencesofgameswherethestartinggameisthe protocolthatismodeledandthefinalgameisanidealprotocolwherethesecurityisgiven byconstruction.Todeemaprotocolsecure,thesequenceofgamesmustresultfromsmall, consecutivemodificationsthataremainlyunnoticeableuptoacertainnegligible probability.Theprobabilityofnoticingdivergencesbetweengamesisarguedbasedonthe securityofacomponentusedintheprotocol(e.g.,theunforgeabilityofasignature scheme).Therefore,theoutputofCryptoVerifwhentheprotocolissecureisaprobability boundingtheadversary'sabilitytobreaktheprotocol. ModelingOckamIdentitiesUsingCryptoVerif WeusedCryptoVeriftomodelsomeaspectsofidentitiesinOckam.Modelingwith CryptoVerifismuchmoretimeconsuming.Therefore,wemadeseveralsimplificationsin ourmodel.Themodelbelowanalyzesthesecurityofidentitiesuponcreationandupon creationofthenextprimarykey. hasheye 32OckamDesignReview PUBLIC

Atahighlevel,weconsideranexisting(honest)identity,andwedemandthattheadversary notbeabletoclaimthatidentity.Furthermore,wealsodemandthattheadversarymay notproduceasubsequentprimarykeythatwillbeaccepted.Inotherwords,theadversary shouldnotbeabletocreateavalidsignaturefortheinitialchangedataassociatedwiththe identitynorforthesubsequentprimarykey.AswediscussedinTOB-OCK-6,aninsightfrom thismodelisthatavalidsignatureonachangedoesnotimplythatitwascreatedusingthe correspondingprivatekeyunlessthesignatureschemeisSUF-CMAsecure. Wedescribethemodelbelow.First,wedefineparametersandthecustomtypesusedin themodel(figureC.2.1). (*Parameters*) typekeyseed[large,fixed].(*Typeforseedusedforkeygeneration*) typepkey[bounded]. (*Typeforpublickey*) typeskey[bounded].(*Typeforsecretkey*) typesignature[bounded]. (*Typeforsignatures*) typehashoutput[fixed,large].(*Typeforhashoutput*) typehashkey_t[fixed]. (*TypeforhashkeytomodelarandomOracle*) (*NILconstantinputinthefirstchangedata*) constNIL:hashoutput. FigureC.2.1:Typedefinitionsandconstants(ockam-identities.ocv:5-14) Next,wedefinethecryptographicprimitivesweuse—namely,asignatureschemeanda hashfunction.Toexperimentwithdifferentassumptionsforthesignaturescheme,wemay replacethe SUF_CMA_det_signature macrowiththe UF_CMA_det_signature macro.In thesecondcase,weassumeonlyEUF-CMAsecurity. (*Ahashfunctionmodeledasarandomoracle*) expandROM_hash_2(hashkey_t,hashoutput,pkey,hashoutput,RO,RO_Oracle,qRom). (*Signatures*) probaPsign.(*Breakingthe(S)UF-CMAproperty*) probaPsigncoll. (*Collisionbetweenindependentlygeneratedkeys*) expandSUF_CMA_det_signature(keyseed, (*Seedusedforkeygeneration*) pkey,(*Publickey*) skey,(*Secretkey*) hashoutput, (*Inputspaceofsignaturescheme*) signature,(*Signature(output)spaceofsignaturescheme*) skgen, (*Secretkeygenerationalgorithm*) pkgen,(*Publickeygenerationalgorithm*) sign,(*Signaturealgorithm*) verify,(*Verificationalgorithm*) Psign, Psigncoll). FigureC.2.2:Definitionofcryptographicprimitives(ockam-identities.ocv:16-34) hasheye 33OckamDesignReview PUBLIC

Next,wedefinehelperfunctions. (*Helperfunctionstoserializechangedata*) funserialize_change_1(pkey,pkey,hashoutput,signature,signature,signature): bitstring[data]. (*Helperfunctiontosimplifysignaturekeygeneration*) letfunkeygen()= rk←Rkeyseed; sk←skgen(rk); pk←pkgen(rk); (sk,pk). FigureC.2.3:Serializationandkeygenerationhelpers(ockam-identities.ocv:36-43) Thenwedefineeventsandaquerythatcapturesthesecuritypropertythatwewantto analyze—namely,theunforgeabilityofchangehistories. (* Eventsandcorrespondingqueriescapturethedesiredpropertiesoftheprotocol. Weconsidertwousers,A&B.BknowsA'sidentityandengagesinaprotocolwith A,whereAprovesknowledgeoftheinitialprimarypublickey.Inasecond interaction,Ageneratesanewprimarypublickeyandsendsthecompletechange historytoB. `make_change_0`issettotruewhenAgeneratesasignatureovertheinitial changedata *) eventmake_change_0(pkey,signature). (* `make_change_1`issettotruewhenAgeneratesasignatureoverthenewchange data *) eventmake_change_1(pkey,pkey,hashoutput,signature,signature,signature). (*`make_change_0`issettotrueifBacceptstheinitialprimarykey*) eventaccept_change_0(pkey,signature). (*`make_change_1`issettotrueifBacceptsthenewprimarykey*) eventaccept_change_1(pkey,pkey,hashoutput,signature,signature,signature). (* Thefirstqueryassertsthatifbacceptstheinitialchange,thenAmusthave generatedthesignaturethatBaccepted.Inotherwords,any`accept_change_0` mustcorrespondtoa`make_change_0`event.Thequeryis*injective*,meaningwe expecteacheventtohappenonceasmodeled. *) querypk:pkey,sig:signature;inj-event(accept_change_0(pk,sig))⟹ inj-event(make_change_0(pk,sig)). (* Thesecondqueryassertsthatifbacceptsthenewchange,thenAmusthave generatedthesignaturethatBaccepted.Inotherwords,any`accept_change_1`

mustcorrespondtoa`make_change_1`event.Thequeryis*injective*,meaningwe expecteacheventtohappenonceasmodeled. *) hasheye 34OckamDesignReview PUBLIC

querypk_0:pkey,pk_1:pkey,change_hash_1:hashoutput,sig_0:signature,sig_1_0: signature,sig_1_1:signature;inj-event(accept_change_1(pk_0,pk_1,change_hash_1, sig_0,sig_1_0,sig_1_1))⟹inj-event(make_change_1(pk_0,pk_1,change_hash_1, sig_0,sig_1_0,sig_1_1)). FigureC.2.4:Eventsandqueries(ockam-identities.ocv:46-73) Nowwedefinetheprocessthatmodelsusers'actions—thatis,thegenerationand verificationofsignedchangehistories. (* `processB`knowsanidentity`change_hash0`.Theadversarycansubmit**in parallel**asignedchangeforeithertheinitialprimarykeyorthenewprimary key. *) letprocessB(hf:hashkey_t,change_hash0:hashoutput,prim_pk_0:pkey)= ((* Breceivesapublickeyandsignatureprovingasinitialchangedata.B verifiesthesignatureandacceptsthenewidentityiftheverification succeeds.Inthatcase,Bsetstheevent`accept_change_0`totrue. *) OinitB(pk:pkey,sig:signature):= change_hash←RO(hf,NIL,pk); ifchange_hash=change_hash0&&verify(change_hash0,pk,sig)then (eventaccept_change_0(pk,sig); return(true)) else (return(false))) | ((* Breceiveschangecorrespondingtoanewchangedata.Bverifiesthesigned historyandsets`accept_change_1`totrueifverificationsucceeds. *) Overfy_change1(new_change:bitstring):= (letserialize_change_1(pk_0,pk_1,change_hash_1,sig_0,sig_1_0,sig_1_1)= new_changein computed_change_hash0←RO(hf,NIL,pk_0); computed_change_hash1←RO(hf,change_hash0,pk_1); computed_change_hash0_is_valid←computed_change_hash0=change_hash0&& verify(change_hash0,pk_0,sig_0); computed_change_hash1_is_valid← computed_change_hash1=change_hash_1&& verify(change_hash_1,pk_0,sig_1_0)&&verify(change_hash_1,pk_1,sig_1_1); ifcomputed_change_hash0_is_valid&&computed_change_hash1_is_validthen (hasheye 35OckamDesignReview PUBLIC

eventaccept_change_1(pk_0,pk_1,change_hash_1,sig_0,sig_1_0,sig_1_1); return(true)) else (return(false)))). FigureC.2.5:Userprocesses(ockam-identities.ocv:101-141) Finally,wedefinethemainprocessthatrunsthegameandexposestheoraclestothe adversary.Wealsodefineahelperfunctionthatgeneratessigningandverificationkey pairs. process Ostart():= (*`hf`isahashkey,selectingahashfunctionatrandom.*) hf←Rhashkey_t; (* Theprimarykeyisgeneratedandthecorrespondingidentity`change_hash0` isalsocomputed *) let(prim_sk_0:skey,prim_pk_0:pkey)=keygen()in change_hash0←RO(hf,NIL,prim_pk_0); return(change_hash0);(*`change_hash0`isreturnedtotheadversary*) (* Thegameexposestheoraclesdefinedby`processA`and`processB`tothe Adversary.Theadversarycanalsoquerytherandomoraclevia`RO_Oracle`. *) (runprocessA(hf,prim_sk_0,prim_pk_0) | runprocessB(hf,change_hash0,prim_pk_0) | runRO_Oracle(hf)) FigureC.2.6:Themainprocess(ockam-identities.ocv:144-164) WecanrunCryptoVerifwiththefollowingcommand: CryptoVerifockam-identities.ocv, Asuccessfulproofshowsthatallquerieshavebeenproved.Theoutputalsocontainsthe probabilitythattheadversarymayinvalidateeachqueryandhencebreakthesecurityof changehistories. hasheye 36OckamDesignReview PUBLIC

Atahighlevel,theresultbelowshowsthatbreakingthesecurityofchangehistoriesis possibleonlywithasmallprobabilityrelatedtoforgingsignaturesofasecuresignature schemeandtheprobabilityofcreatinghashcollisions. [...] RESULTProvedforall sig_1_1,sig_1_0_0,sig_0_0:signature, change_hash_1:hashoutput, pk_1,pk_0_0:pkey; inj-event(accept_change_1(pk_0_0,pk_1,change_hash_1,sig_0_0,sig_1_0_0,sig_1_1))⟹ inj-event(make_change_1(pk_0_0,pk_1,change_hash_1,sig_0_0,sig_1_0_0,sig_1_1)) uptoprobability Psign(time_2,1)+Psign(time_1,2)+(1.5*qRom^2+29+12*qRom)/ |hashoutput|+10*Psigncoll [...] RESULTProvedforall sig:signature, pk:pkey; inj-event(accept_change_0(pk,sig))⟹inj-event(make_change_0(pk,sig)) uptoprobability Psign(time_1,2)+(1.5*qRom^2+10*qRom+26)/ |hashoutput|+7*Psigncoll [...] FigureC.2.7:CryptoVerifoutputexcerpt Here,weusedtherandomoraclemodelforeaseofmodeling;however,amuchweaker collisionresistancepropertywouldsufficetoprovetheresult. hasheye 37OckamDesignReview PUBLIC

D.UseCaseDiagrams Thefollowingdiagramsshowthevariousstepstakenforthe TCPPortalsandKafkaPortals usecases,includingtherelationshipsbetweenvariouskeysandstorageitems.SectionD.1 containsgeneraldiagramsthatshowprocessescommontobothusecases,whilesection D.2containsdiagramsforthe TCPPortalsusecase,andsectionD.3containsdiagramsfor the KafkaPortalsusecase. D.1General Overview hasheye 38OckamDesignReview PUBLIC

ProjectNodeHierarchy hasheye 40OckamDesignReview PUBLIC

CreateSpaceandProject hasheye 41OckamDesignReview PUBLIC

GetProjectCredentials GetProjectTicket hasheye 42OckamDesignReview PUBLIC

D.2TCPPortals OutletandRelayCreation InletCreationandPortal hasheye 43OckamDesignReview PUBLIC

D.3KafkaPortals KafkaConfig KafkaConsumerCreation hasheye 44OckamDesignReview PUBLIC

KafkaProducerCreationandPortal hasheye 45OckamDesignReview PUBLIC