Protokollbeskrivning av OKI

1 Syfte

Det här dokumentet har som syfte att beskriva protokollet OKI.

2 Sammanfattning

OKI är tänkt som en öppen standard för kryptering och signering av information.

3 Klassificering

Dokumentet har klassificerats som öppet dokument (Klass A).

4 Innehåll

5 Översikt

Protokollet kan användas för att skicka krypterade meddelanden/filer/dokument eller digitalt signera filer/dokument. I protokollet finns två aktörer inblandade: användare och CA. Användarna är de som ska skicka krypterad information mellan varandra och ett CA är en organisation som intygar användarnas identitet.

Varje användare måste först generera ett eget nyckelpar. Paret består av två olika nycklar, en privat nyckel och en publik nyckel. Den publika nyckeln kan bara användas för att kryptera information eller för att verifiera en digital signatur, och den behöver därför inte hållas hemlig. Den privata nyckeln däremot är den som används för att dekryptera den information som krypteras med den privata nyckeln, eller för att skapa de signaturer som kan verifieras med den publika nyckeln. Den privata nyckeln måste därför lagras hemligt.

När ett nyckelpar har genererats måste varje användare skapa ett eget och personligt certifikat. För att skapa ett certifikat bör man ha minst en certifiering av användarens identitet. För att få en certifiering krävs det att en organisation som känner igen dig har startat ett CA och kan intyga din identitet. Du sänder dem din privata nyckel, de verifierar att du är den du påstår och att förfrågan är autentisk, och sedan skickar de tillbaka en certifiering för den här nyckeln. Processen kan sedan upprepas med flera CA om du vill ha fler certifieringar. När du har en certifiering kan du skapa ditt eget certifikat från din publika nyckel och de certifieringar du vill att ska ingå i ditt certifikat. Det går också att skapa certifikat utan certifieringar, men det är inte rekommenderat.

5.1 Verifiering av certifikatet

Certifikatet måste verifieras innan kryptering av information, vid utvärdering av en digital signatur eller vid mottagandet av ett krypterat och signerat meddelande. Användaren som ska kryptera information, eller verifiera en signatur får då ta ställning till om det här är en nyckel som går att lita på. Detta görs genom att varje certifikatet bedöms utifrån de certifieringar som verifieras, och utifrån de betrodda certifieringarna i certifikatet avgörs om användarens identitet kan anses vara bekräftad. Rent praktiskt bör programmet inte visa de certifieringar för användaren som inte kan verifieras av en betrodd publik certifieringsnyckel. På så vis görs en bedömning från den information som är pålitlig.

Om ett certifikat inte innehåller några betrodda certifieringar kvarstår bara en beskrivning av krypteringsnyckel via ett fingerprint. Användaren får då kontakta mottagaren och be om ett nytt och uppdaterat certifikat, eller manuellt verifiera att användaren verkligen använder ett certifikat med detta fingerprint. Ett fingerprint är en unik sifferkod och det kan bara finnas en enstaka krypteringsnyckel med denna sifferkod. Om sifferkoden på nyckeln stämmer mot den som användaren påstår sig använda kan certifikat och nyckeln anses pålitlig.

Ett certifikat kan också vara osignerat utan certifieringar, men det är inte rekommenderat. Vid utvärdering av certifikatet måste det alltid verifieras manuellt via fingerprint.

5.1 Revokering

När ett certifikat utvärderas bör man också kontrollera att det är ett certifikat som fortfarande används av denna användaren eller om det är ett certifikat som är revokerat (återkallat). Rutiner för detta ingår inte i denna standarden. Det är därför viktigt att de certifikat som används alltid är de senaste. Det enklaste är att alltid hämta det senaste certifikatet innan kryptering eller verifiering av signaturer med det här certifikatet. Om det dyker upp fler certifikat för samma användare bör naturligtvis det med senast tid/datum på certifieringarna betraktas som giltigt och de andra förkastas.

Ytterligare rutiner eller hjälpstandarder för att säkerställa att certifikat inte har revokerats kan alltså tillkomma, men vid skrivandets stund anses det inte som en nödvändig del av den här standarden.

6 MStream

All data i protokollet lagras på mstream format. Det betyder att informationen lagras med en av fyra olika informationsprimitiver (*Int8*, *Int16*, *Int32* eller *MData*) som följer efter varandra i en dataström. Varje primitiv har en väl definierad betydelse och datalängd. Det här minskar risken att informationen misstolkas och ska göra det mycket lättare att skriva kod utan sårbarheter.

6.1 Int8, Int16 och Int32

De tre första datatyperna (*Int8*, *Int16* och *Int32*) är positiva heltal (unsigned values) med 8, 16 eller 32 bitars längd. Datatyperna hanteras i programkoden exempelvis med variabeltypen *unsigned int*.

6.2 MData

Den fjärde och sista datatypen är *MData* som används för att lagra en datasträng. Strängen lagras först som ett positivt 32bit heltal format följt av en sträng av denna längden med positiva 8 bitars heltal. Eftersom strängen alltid är definierad med ett inledande tal kan en passande buffert skapas i förväg för att lagra strängen vid tolkning av informationen och den behöver aldrig skriver in i en buffert med fast längd. Detta minskar drastiskt risken för sårbarheter i koden av typen buffer overflows.

6.3 Tolkning

Vid tolkning av mstream formatet används en läspekare som räknas fram vartefter läsningen av informationen sker. Tolkningen av informationen sker efter det förväntade formatet på informationen genom att varje datatyp efterfrågas i den ordning de förväntas från dataströmmen, och inte med utgångspunkt från informationens faktiska innehåll. Detta gör att en korrekt läsning av informationen framtvingas, även om formatet är felaktigt, fast naturligtvis med felaktiga värden på informationsprimitiverna som följd. Det här ska göra det svårt att avsiktligt framtvinga någon annan tolkning av informationen.

Vid tolkning kan information alltid läsas från en dataströmmen, oavsett om den lagrade informationen tar slut vid läsning. Om informationen tar slut innan all information är läst så resulterar resterande läsningar från dataströmmen i värdet 0 på Int8, Int16 eller Int32, eller en tom MData med längd 0 om en MData ska läsas. Om en MData har en längd som är längre än vad det finns information kvar att läsa, läses bara den information som finns tillgänglig. Strängarnas maximala längd begränsas därmed effektivt av informationens längd.

Varje fält i datastrukturen kommer därmed kunna läsas från dataströmmen, men kan ha ett tomt eller felaktigt värde, vilket fortfarande måste kontrolleras vid tolkning av informationen.

7 Dataformat

Protokollet består av 9 väldefinierade dataformat (datastrukturer). Varje dataformat innehåller den information som är nödvändig för att protokollet ska fungera. Alla dataformaten har mstream format.

- PublicKey
- PrivateKey
- Signature
- EncryptedKey
- EncryptedData
- Certification
- Certificate
- PublicCertKey
- PrivateCertKey

7.1 PublicKey

PublicKey är ett datastruktur som innehåller en användares publika krypteringsnyckel. Denna nyckeln kan spridas oskyddat eftersom nyckeln är publik, men kan inte användas för kryptering eftersom den inte är certifierad ännu.

MData **IDTag ”PublicKey”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

Int16 **EncrKeyType**
0 = ENCR_RSA_2048
1 = ENCR_RSA_3072
2 = ENCR_RSA_4096

Int16 **SignKeyType**
0 = SIGN_RSA_2048
1 = SIGN_RSA_3072
2 = SIGN_RSA_4096

MData **PublicEncrKey**
En publik krypteringsnyckel för kryptering.

MData **PublicSignKey**
En publik krypteringsnyckel för verifiering av signaturer.

7.2 PrivateKey

PrivateKey innehåller användarens privata nyckel som används vid dekryptering och signering. Denna datastruktur ska lagras på skyddad plats, exempelvis skyddad med kryptering.

MData **IDTag ”PrivateKey”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

Int16 **EncrKeyType**
0 = ENCR_RSA_2048
1 = ENCR_RSA_3072
2 = ENCR_RSA_4096

Int16 **SignKeyType**
0 = SIGN_RSA_2048
1 = SIGN_RSA_3072
2 = SIGN_RSA_4096

MData **PrivateEncrKey**
En privat krypteringsnyckel för dekryptering.

MData **PrivateSignKey**
En privat krypteringsnyckel för signering.

7.3 Signature

Signature innehåller en signatur över en digital datamängd (ex fil, meddelande). Signaturen knyter användaren som utfärdat signaturen till det signerade innehållet. Om innehållet ändras bara en enstaka bit blir signaturen ogiltig. Datamängden lagras inte i denna datastruktur utan skickas separat från signaturen.

MData **IDTag ”Signature”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

Int16 **DigestType**
0 = DIGEST_SHA_256

MData **SignInfoData**
En inre datastruktur (se nedan) som beskriver signaturen ytterligare. Denna structen ingår även i den signerade informationen.

MData **Signature**
En digital signatur över en datamängd. Signaturens typ och längd avgörs av den nyckel som användes vid signering.

SignInfoData har följande format:

MData **SignTag**
En tag som beskriver tillämpningen av signaturen för att inte blanda samman signaturer mellan olika tillämpningar. Vid signering av EncryptedKey används taggen ”EncryptedKeySiganture”.

Int32 **TimeStamp**
En tidsstämpel för när informationen upprättades, eller värdet 0 om ingen tidsstämpling valts. Tiden har formatet UNIX Epoc.

MData **SignersMessage**
Ett valfritt meddelande från användaren. Ex ”Har granskat dokument och det ser bra ut.”

7.4 EncryptedKey

EncryptedKey innehåller en krypterad sessionsnyckel. Sessionsnyckeln kan användas för kryptering, autentisering eller någon annan kryptografisk operation. I ett meddelande med fler mottagare kan informationen krypteras med flera sessionsnycklar, en för varje användare.

MData **IDTag ”EncryptedKey”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

MData **EncryptedKey**
En asymmetriskt krypterad sessionsnyckel. Informationens typ och längd avgörs av den nyckel som användes vid kryptering.

7.5 EncryptedData

EncryptedData innehåller en mängd krypterad och signerad information. Informationen är krypterad till en specifik användaren och kan bara dekrypterad med denna användarens privata krypteringsnyckel. Signaturen gör det möjligt att säkerställa att informationen kommer från sändaren.

MData **IDTag ”EncryptedData”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

Int16 **EncryptionType**
0 = AUTH_ENCR_AES_256_CBC_HMAC_SHA256

Int16 **SignDigestType**
0 = DIGEST_SHA_256

MData **EncryptedSessionKey**
En asymmetriskt krypterad sessionsnyckel. Informationens typ och längd avgörs av den nyckel som användes vid kryptering.

MData **Signature**
En digital signatur över EncryptedSessionKey. Vid signering används en digest av typen som anges av SignDigestType. Signaturens typ och längd bestäms i övrigt av den nyckel som skapade signaturen.

MData **EncryptedData**
Information krypterad med sessionsnyckel ovanför. Vilken symmetrisk krypteringsalgoritm som använts avgörs av parametern ovanför. Krypteringen kombineras alltid med autentisering.

7.6 Certification

Certification innehåller en certifiering. En certifiering är ett intyg (digital signatur) från ett CA som intygar att en privat nyckel tillhör en viss identitet. En eller flera certifieringar kan sedan användas för att skapa ett certifikat för en användare. Certifieringen innehåller inte användaren privata nyckel utan är bara ett intyg att en viss nyckel tillhör en identitet. Certifieringen kan inte användas fkr kryptering eller verifiering av signaturer.

MData **IDTag ”Certification”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

MData **KeyID**
Ett unikt nummer (64bit sträng) för den certifieringsnyckel som användes vid certifieringen. Numret ska göra det lättare att hitta rätt publika certifieringsnyckel.

MData **OrgName**
Namnet på den organisation som påstås ha denna nyckeln. Även detta värde är till för att hitta rätt nyckel om den saknas.

MData **UserInfoData**
En inre datastruktur (se nedan) som beskriver vilken användare som har certifierats.

MData **CertSign**
En signatur över den publika krypteringsnyckeln och datastrukturen **UserInfoData** utfärdat av en certifieringsnyckel. Formatet och längden på signaturen beror på vilken nyckel som använts vid signering.

MData **PubKeyFingerprint**
Ett fingerprint för den publika nyckeln som certifieringen gäller för.

UserInfoData har följande format:

Int32 **TimeStamp**
En tidsstämpel för när informationen upprättades, eller värdet 0 om ingen tidsstämpling valts. Tiden har formatet UNIX Epoc.

Int16 **FealdCount**
Antal datafält som följer nedanför.

För varje datafält: {

MData **FealdName**
En inledande beskrivning av datafältets innehåll. Ex ”namn”, ”e-post”, etc.

MData **FealdData**
Motsvarande data för det här datafältet.

}

7.7 Certificate

Certificate innehåller ett certifikat för en användare. Certifikatet kan användas för kryptering av information till den här användaren eller för verifiering av signaturer utfärdade av användaren. Certifikatet består av en publik krypteringsnyckel följt av flera certifieringar. Varje certifiering innehåller en beskrivning av användarens identitet och kan innehålla olika typ av information.

MData **IDTag ”Certificate”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

MData **UserPubKey**
Användarens publika nyckel på formatet **PublicKey**.

Int16 **CertificationCount**
Antal certifieringar i certifikatet (kan ha värdet 0).

För varje certifiering: {

MData **KeyID**
Nyckel-id (64bit sträng) för certifieringsnyckeln som signerat certifieringen.

MData **OrgName**
Namnet på den organisation som påstås skapat nyckeln som signerat.

MData **UserInfoData**
En inre datastruktur som beskriver vilken användare som har certifierats (för format, se **Certification**).

MData **CertSign**
Signatur över den publika nyckeln och **UserInfoData** ovanför.

}

7.8 PublicCertKey

PublicCertKey innehåller en publik certifieringsnyckel. Den publika certifieringsnyckeln används för att verifiera certifieringar utfärdade av ett CA. Den publika certifieringsnyckeln ska publiceras på minst en offentlig plats, exempelvis en hemsida.

MData **IDTag ”PublicCertKey”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

Int16 **KeyType**
0 = SIGN_RSA_2048
1 = SIGN_RSA_3072
2 = SIGN_RSA_4096

MData **KeyID**
En 64bit (8 bytes) lång slumpmässig sträng för att identifiera denna nyckel.

MData **OrgName**
En sträng som beskriver namnet på den organisation som har skapat nyckeln.

MData **PublicSignKey**
En publik signeringsnyckel för verifiering av signerade certifieringar.

7.9 PrivateCertKey

PrivateCertKey innehåller en privat certifieringsnyckel som används för att utfärda certifieringar för publika nycklar. Nyckeln kan inte användas för kryptering/dekryptering utan bara för signering av användares publika nycklar. Den privata nyckeln måste naturligtvis hanteras skyddat och kan exempelvis lagras på ett separat datorsystem avskilt från omvärlden.

MData **IDTag ”PrivateCertKey”**
En MData med ASCII-strängen ovanför för att identifiera formatet.

Int16 **Version**
Versionsnummer på datastrukturen. Ska alltid ha värdet 0 i första versionen.

Int16 **KeyType**
0 = SIGN_RSA_2048
1 = SIGN_RSA_3072
2 = SIGN_RSA_4096

MData **KeyID**
En 64bit (8 bytes) lång slumpmässig sträng för att identifiera denna nyckel.

MData **OrgName**
En sträng som beskriver namnet på den organisation som har skapat nyckeln.

MData **PrivateSignKey**
En privat signeringsnyckel för utfärdande av nya certifieringar.

8 Nycklar och algoritmer

De här stycket definierar de algoritmer som förekommer i standarden.

8.1 Asymmetriska Krypteringsalgoritmer

Tre krypteringsalgoritmer används av standarden.

ENCR_RSA_2048
ENCR_RSA_3072
ENCR_RSA_4096

Algoritmerna är definierade på det sätt de är implementerade i ”OKI Encrypt and Sign Tool”.

8.2 Signeringsalgoritmer

Tre olika signeringsalgoritmer används av standarden.

SIGN_RSA_2048
SIGN_RSA_3072
SIGN_RSA_4096

Algoritmerna är definierade på det sätt de är implementerade i ”OKI Encrypt and Sign Tool”.

8.3 Symmetriska Krypteringsalgoritmer

Bara en symmetrisk krypteringsalgoritm används av standarden.

AUTH_ENCR_AES_256_CBC_HMAC_SHA256

Algoritmen är definierade på det sätt den är implementerade i ”OKI Encrypt and Sign Tool”.

8.4 Message Digest Algoritmer

Bara en message digest algoritm används av standarden.

DIGEST_SHA_256

Algoritmen är definierade på det sätt den är implementerade i ”OKI Encrypt and Sign Tool”.