Norwegian Tax Administration

Requirements and guidelines for implementing digital signatures in Cash Register Systems

Revision released July 2017

Table of Contents

1 Introduction

In order to achieve compliance with the Norwegian Cash Register Systems Act and Regulations pursuant to the Act all Cash Register Systems are required to implement a signature. There is, however, an exemption to this rule. See the following statement for more information.

The signature shall sign specific data from each receipt and be recorded in the electronic journal upon finalization of each transaction. It is also mandatory to export the signature to the SAF-T Cash Register XML.

For the creation of the signature the system vendors need to choose one of two alternatives:

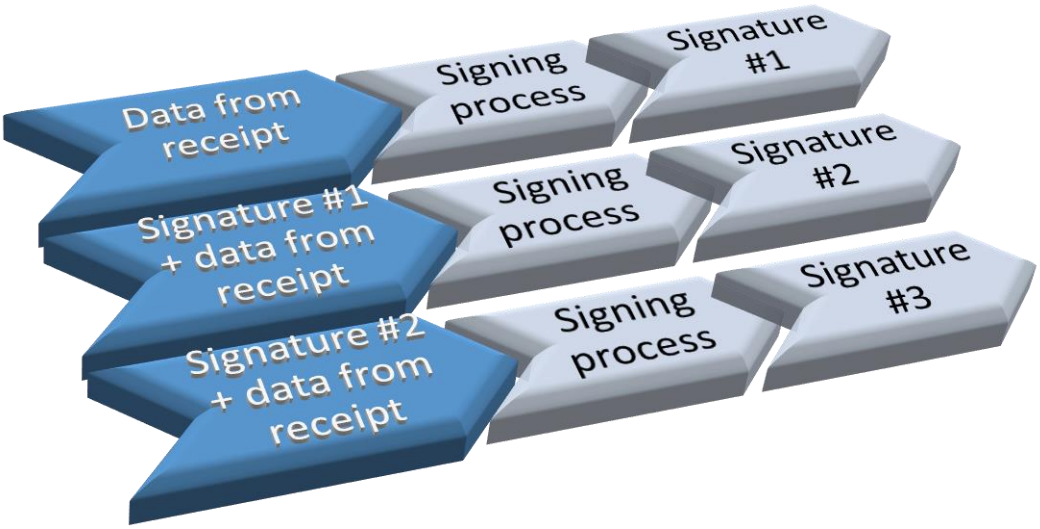
- A digital signature using an RSA 1024 bit key with a SHA-1 hash function (RSA-SHA1-1024)
- A keyed-hash message authentication code using a 128 bit key with a SHA-1 hash function (HMAC-SHA1-128)

The vendor must use RSA-SHA1-1024 as long as it is applicable, and only use HMAC-SHA1-128 if RSA is not feasible. This assessment must be made for each of the systems to be declared.

This document explains the basic principles of these two alternatives, and provides a step-by-step guide to implementing RSA-SHA1-1024 and HMAC-SHA1-128. It further presents best practices regarding key generation and key management.

1.1 The main principles

The figure presents an overview of the process of signing data:



Data from receipt is defined in section 2. The signing of data is done using RSA-SHA1-1024 or HMAC-SHA1-128, which return the signatures.

The use of signatures from the previous receipt ensures a chain of signed data that should not be broken. This must be done in the same ECR or Point of sale (POS) or other logical representation of the point of sale (i.e. terminal id etc.).

The signing process MUST be done during the completion of a transaction, not by batch processing etc.

1.2 Legal basis for the signature

With reference to:

Regulations no. 1616 of 18.12.2015 relating to requirements for cash register systems (the Cash Register Systems Regulations) - Section 2-7 Electronic journal – (1)

- *"The electronic journal must be protected against alteration and deletion."*

Comments on the Cash Register Systems Regulations – Regarding Section 2-7 Electronic journal:

- *"In order for an electronic journal to be securely protected against alteration, it must not be possible to directly or indirectly manipulate the contents of the journal. The cash register system must therefore have an electronic journal which, in practice, is invisible to the user and preferably locked with a digital signature and/or encryption."*

With the signature, any alteration of the signed data without use of the private key of the software vendor is made detectable. This adds a strong integrity measure to the electronic journal. However, it does not protect the journal from being altered or deleted.

The statement will also describe the responsibilities of the software vendor, and the responsibilities of their customers who are obliged to keep their accounting data from cash sales.

1.3 Cryptography

Two basic techniques for encrypting information:

- Asymmetric cryptography, also called public key cryptography
- Symmetric cryptography, also called secret key cryptography

Asymmetric cryptography is a cryptographic system that uses two related keys, a key pair: *private key* and *public key.* Any message that is encrypted by using the private key can only be decrypted using the matching public key. The public key is made freely available to anyone, while a second, private key is kept secret and only known to the vendor.

Symmetric cryptography on the other hand uses the *same key* (*secret key*) for both encryption and decryption. As long as both sender and recipient know the secret key, they can encrypt and decrypt all messages that use this key.

The advantage to asymmetric cryptography is that it eliminates the issue of exchanging keys over the Internet or large network while preventing them from falling into the wrong hands. The disadvantage is that it is slower than symmetric encryption. It requires more processing power to both encrypt and decrypt the content of the message.

1.4 Digital signature

Digital signatures are based on asymmetric cryptography. Using a public key algorithm such as RSA, two keys that are mathematically linked are generated: one private and one public. To generate a digital signature, signing software creates a one-way hash (such as SHA-1) of the electronic data to be signed. The private key is then used to encrypt the hash. The encrypted hash is the digital signature. Digital signature provides integrity, authentication and non-repudiation.

1.5 Keyed-hash Message authentication Code

A keyed-hash message authentication code (HMAC) is a specific type of Message Authentication Code (MAC) and is based on symmetric cryptography. The HMAC process mixes a secret key with the message data, hashes the result with the hash function (such as SHA-1), mixes that hash value with the secret key again, and then applies the hash function a second time. An HMAC can be used to determine whether a message sent over an insecure channel has been tampered with, provided that the sender and receiver share a secret key. HMAC provides integrity and authentication, but not non-repudiation as the secret key is known by more than one entity.

1.6 Validation support

The Norwegian Tax Administration will develop a tool for system vendors to test single digital signatures for both the RSA and HMAC method. Expected released is during autumn of 2017.

2 Technical requirements

This section addresses the technical requirements for implementing RSA or HMAC.

2.1 General requirements

1. The signature must be recorded in the electronic journal with a direct link to the full record of the original receipt.
2. The signature must be created for all transactions that are reported in **cashtransaction**. This includes all receipts that are given a transaction number, and normally comprises transactions that influence sales. Please see section 2.1.2 for further clarification of what types of transactions this includes.
3. It must be recorded which version of the private or secret key that was used to generate the signature of the receipt.
4. The format for creating the hash and signature must be identical to the data exported to the Cash Register XML format and is stated in section 2.2.4 (RSA) or section 2.3.3 (HMAC). The data elements must be separated by ";" (semicolon). Further, the currency must be the same as stated on the receipt.
5. The signature must be created and recorded in the electronic journal in parallel with finalizing the transaction. Not by batch processing.
6. The signature from previous receipt must be derived from the signature value from the last receipt for the same company in the same cash register. See example in section 2.1.1.
7. When the previous receipt does not have a signature, for example after a fresh install, the signature value must be set to "0" – number zero.
8. When exporting from the electronic journal to Cash Register XML the signature shall be recorded in the field 'signature' and the key version in the field 'keyVersion'. Both fields are located in company/location/cashregister/cashtransaction.

2.1.1 Example signature trail

The signature trail must follow the structure of the SAF-T Cash Register XML. This means that the signature for a receipt must have the data from that receipt included, as well as the signature from the previous receipt (receipt number) for the same company in the same cash register.

When exporting to the XML datafile the receipts must be in the same order as shown below. This is to make it possible to verify the signature trail.

Company1
- Location1
  - CashRegisterX
    - Transaction1X1
      - Signatur1X1
    - Transaction1X2
      - Signature1X2 (from Signature1X1)
    - Transaction1X3
      - Signature1X3 (from Signature1X2)
  - CashRegisterY
    - Transaction 1Y1
      - Signature1Y1
    - Transaction1Y2
      - Signature1Y2 (from Signature1Y1)
    - Transaction1Y3
      - Signature1Y3 (from Signature1Y2)

Company2
- Location1
  - CashRegisterX
    - Transaction2X1
      - Signature2X1
    - Transaction2X2
      - Signature2X2 (from Signature2X1)
    - Transaction2X3
      - Signature2X3 (from Signature2X2)

2.1.2 Transactions to include in signature trail

As the signature trail must follow the structure of the SAF-T Cash Register XML, **all transactions** that are exported to auditfile>company>location>cashregister>**cashtransaction** must be given a signature. They must therefore always be included in the signature trail. Due to this fact, the elements signature and keyVersion are mandatory.

Normally the transactions reported in the **cashtransaction** influence sales, both increase and decrease the sales amount. However, depending on the preferences of the system vendor, also other transactions (transaction types) may be included in the signature trail.

If for example “Opening of cash drawer” is treated as a transaction by the system, and used for generation of signature and written as a transaction in the XML export (in <cashtransaction>) this must in addition be reported as an event in the XML with a reference to the transaction ID <transID>.

All fields required to create the signature are mandatory, and they must all be included in the cashtransactions along with the rest of the mandatory elements.

When different types of transactions are used, <transType> is to be filled out to distinguish the different transaction types. The elements <transAmntIn> and <transAmntEx> must be filled with value “0.00” when there is no amount. These elements cannot be left empty or excluded.

2.2 RSA-SHA1-1024

2.2.1 General description and background of RSA

RSASHA1-128 is constructed from the SHA-1 hash function and used as an RSA digital signature. The sender first applies a hash function to the message to create a message digest. The sender then encrypts the message digest with the sender's private key to create the sender's personal signature.

Upon receiving the message and signature, the receiver decrypts the signature using the sender's public key to recover the message digest and hashes the message using the same hash algorithm that the sender used.

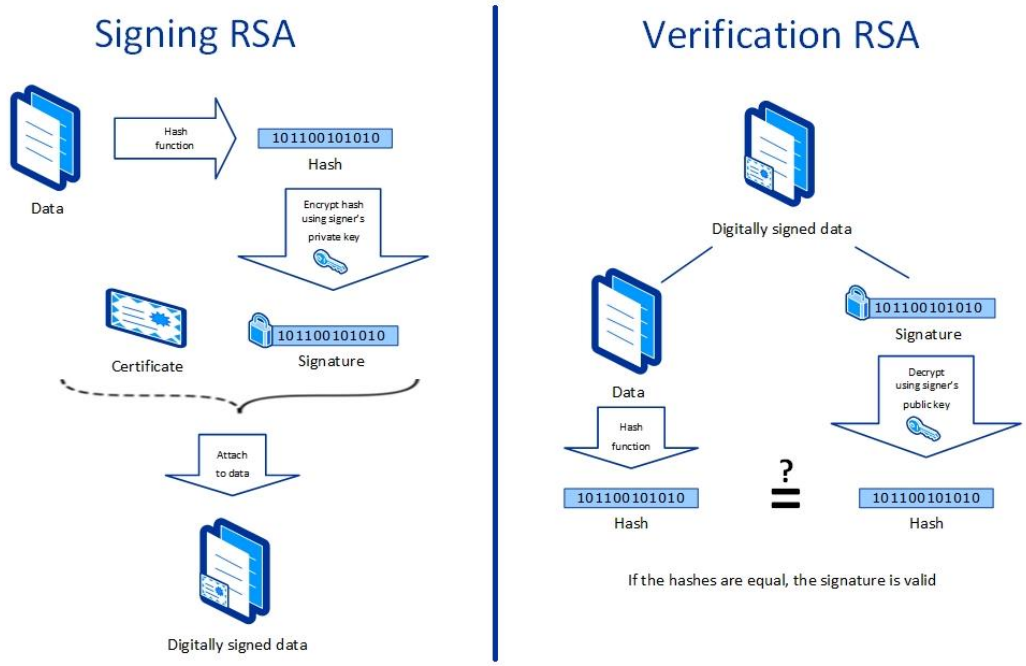
If the message digest that the receiver computes matches the message digest received from the sender, the receiver can assume that the message was not altered while in transit. Note that anyone can verify a signature, because the sender's public key is common knowledge. This technique does not retain the secrecy of the message; for the message to be secret, it must also be encrypted.

2.2.2 Implementation for ECR/POS software of RSA:

The public key must be shared with the Tax Authorities. During audits the Tax Authorities will regenerate the hash and compare it with the hash derived from the signature stored in the electronic journal by using the public key. The RSA signature must be generated and stored in the electronic journal upon each completion of the sale.

The Tax Authorities generates the hash by using the defined data values stored in the electronic journal. A match indicates that the data has not been altered by a third party without knowledge to the private key.

The process of signing and verification is illustrated in the image below:



2.2.3 Certificate creation

The key pair used does not require the issuance of a certificate by an accredited entity. The systems vendor can generate the self-signed certificate for certification and send the public key to the Norwegian Tax Administration through the Cash Register System product declaration.

To create the certificate from the private key, RSA algorithm must be used with following specifications on the parameters:

- Format = x.509
- Charset = UTF-8
- Encoding = Base-64
- Endianness = Little Endian
- OAEP Padding = PKCS1 v1.5 padding
- Private key size = 1024 bits
- Hash Message Format = SHA-1

2.2.4 Technical requirements

IMPORTANT NOTE: *The process of verifying the signature will not be possible if the technical requirements are not met. This is because the recalculation of the SHA-1 hash must be done with the exact same values as done with the signing process.*

When using the RSA algorithm (data encryption algorithm using the asymmetric key system, public and private key), the following guidelines (in addition to general requirements) must be applied:

1. The systems vendor is not allowed to use different key pairs for the same Cash Register System, for example one key pair for each customer using the same system.

2. The public key must result from an extraction from the private key in PEM format (base-64) and must be submitted to the Norwegian Tax Authorities along with the product declaration for the Cash Register System. Use form RF-1348 in the Altinn portal for this purpose.
3. The systems vendor must ensure that the private key used to create the signature is their unique knowledge and is properly protected in the software environment. See Key Management for further guidelines.

The following table describes what data to be signed and their order:

Element in SAF-T Cash Register	Description of element	Format and requirements	Examples
signature	Signature from previous receipt	Base-64 If no previous receipt the signature value must be set to "0" – number zero	signature_from_previous_receipt
transDate	Date at which the transaction was performed.	YYYY-MM-DD Do NOT use time zone or combined date and time format.	2014-01-24 2015-10-29
transTime	Time at which the transaction was performed.	hh:mm:ss Use ss=00 as default value if no information of seconds are available. Same as within the SAF-T export. Do NOT use time zone or combined date and time format.	23:59:59
nr	Transaction number. This must be a unique, sequential number within a journal. This will be the same as the number stated on the issued receipt	No leading or ending spaces.	123456789
transAmntIn	The amount involved in the transaction, including VAT.	Decimal Data Type: Numerical field with two decimals. Decimal separator "." (dot). No thousand separators. No leading or ending spaces.	1250.00 -1250.00 0.00
transAmntEx	The amount involved in the transaction, excluding VAT.	Decimal Data Type:	1000.00 -1000.00

		Numerical field with two decimals. Decimal separator "." (dot). No thousand separators. No leading or ending spaces.	0.00

2.2.5 Practical example

Signing of data:

Data to be signed:

signature_from_previous_receipt;2014-01-24;23:59:59;123456789;1250.00;1000.00

Hash data with SHA-1:

55abb153c38026540a601dd7bd4302e41481cd37

Encrypt the hash value with private key and generate signature:

signature_for_this_receipt

Verification of signature:

Generate hash with SHA-1 with same data as signed above:

55abb153c38026540a601dd7bd4302e41481cd37

Use public key on signature to get hash:

55abb153c38026540a601dd7bd4302e41481cd37

The hash values are the same and the data integrity is confirmed.

2.2.6 The use of SHA-1

The Tax Authorities are aware of that SHA-1 per definition is cracked (collisions found).

However, the purpose of the hash function in RSA is only to generate a hash string of known data. The signature is within the RSA key pair. Therefore, SHA-1 is not of importance to the RSA signing mechanism used to verify the integrity. The use of SHA-1 with HMAC does not influence the integrity, as the (secret) key is the factor that provides integrity.

The software vendors were presented the use of SHA-1 from the early stages of this implementation. Furthermore, there are no known weaknesses to the use of SHA-1 in combination with RSA and HMAC for the purpose as described in this document.

2.3 HMAC-SHA1-128

2.3.1 General description and background of HMAC:

HMAC-SHA1-128 is a type of keyed hash algorithm that is constructed from the SHA1 hash function and used as an HMAC. The HMAC process mixes a secret key (128 bit) with the receipt data, hashes the result with the hash function, mixes that hash value with the secret key again, and then applies the hash function a second time. The output hash is 160 bits in length.

An HMAC can be used to determine whether a message (data) sent over an insecure channel has been tampered with, provided that the sender and receiver share a secret key. The sender computes the hash value for the original data and sends both the original data and hash value as a single message. The receiver recalculates the hash value on the received message and checks that the computed HMAC matches the transmitted HMAC.

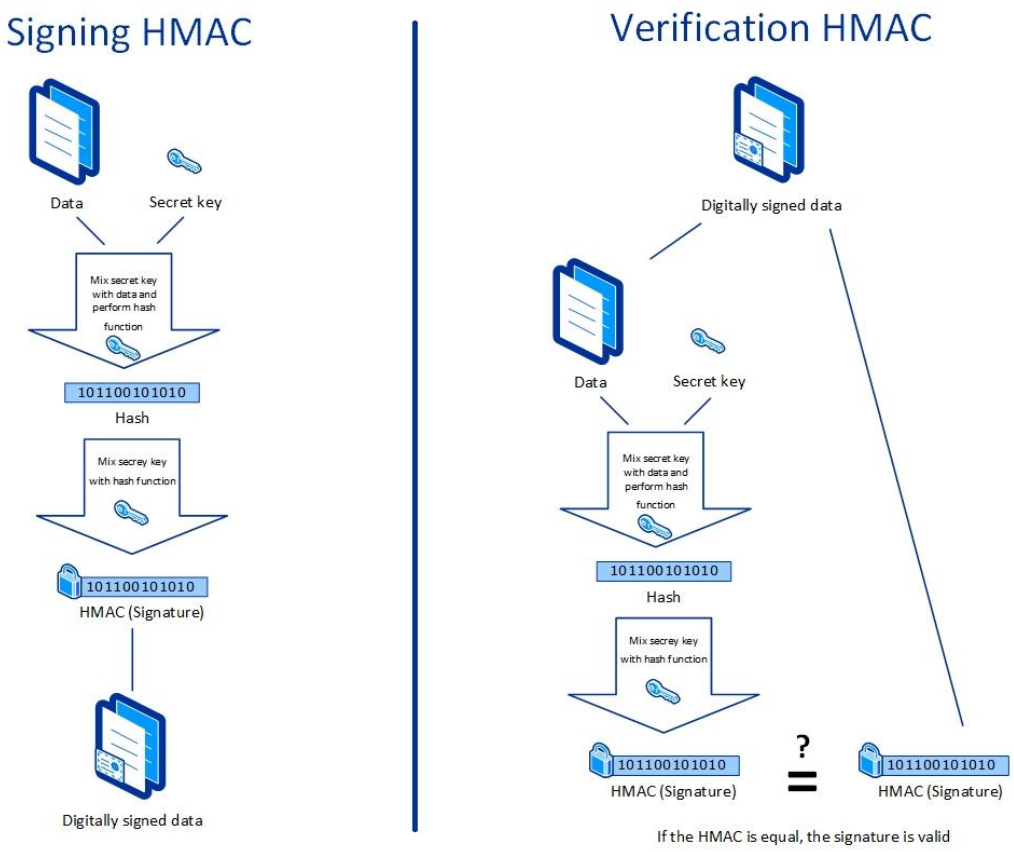
Any change to the data or the hash value results in a mismatch, because knowledge of the secret key is required to change the message and reproduce the correct hash value. Therefore, if the original and computed hash values match, the message is authenticated.

2.3.2 Implementation for ECR/POS software of HMAC:

The secret key must be shared with the Tax Authorities. During audits, the Tax Authorities will regenerate the HMAC and compare it with the HMAC stored in the electronic journal. The HMAC must be generated and stored in the electronic journal upon each completion of the sale, as stated in section 2.1.

The Tax Authorities generates the HMAC by using the defined data values stored in the electronic journal, as well as the secret key. A match indicates that the data has not been altered by a third party without knowledge to the secret key.

The process of both the ECR/POS-software generation of HMAC and integrity verification done by the Tax Authorities is illustrated in the image below:



Definition of HMAC as stated in RFC 2104

$$HMAC(K,m) = H\Big((K' \oplus opad) \mid\mid H\big((K' \oplus ipad) \mid\mid m\big)\Big)$$

Two basic steps when creating HMAC:

1. H(K XOR ipad, receipt_data) = Hash
    a. The HMAC process mixes a secret key (128 bit) with the receipt data, hashes the result with the hash function
2. H(K XOR opad, Hash) = HMAC (Signature)
    a. Mixes that hash value with the secret key again, and then applies the hash function a second time.

Definitions to be used are:

H = Hash using SHA-1
K = Secret key = 128 bits (16 bytes)

B = Data block byte length = 512 bits (64 bytes)
L = Output size of SHA-1 Hash = 160 bits (20 bytes)
M = data from receipt, see table below (+ HMAC from previous receipt) = 160 bits (20 bytes)
ipad = Inner padding = the byte 0x36 repeated B times
opad = outer padding = the byte 0x5C repeated B times
Charset = UTF-8
Encoding = Base-64
Endianness = Little Endian

Please see RFC 2104 for further definitions.

2.3.3 Technical requirements

IMPORTANT NOTE: *The process of verifying the HMAC will not be possible if the technical requirements are not met. This is because the recalculation of HMAC must be done with the exact same values.*

When using the HMAC algorithm (data encryption algorithm using the symmetric key system, with secret key), the following guidelines (in addition to general requirements) must be applied:

1. The systems vendor is not allowed to use different key pairs for the same Cash Register System, for example one key pair for each customer using the same system.
2. The secret key (base 64) must be submitted to the Norwegian Tax Authorities along with the product declaration for the Cash Register System. Use form RF-1348 in the Altinn portal for this purpose.
3. The systems vendor must ensure that the secret key used to create the HMAC is their unique knowledge and is properly protected in the software environment. See Key Management for further guidelines.

The following table shows the data to be signed with HMAC:

Element in SAF-T Cash Register	Description of element	Format and requirements	Examples
signature	HMAC (signature) from previous receipt	Base-64 If no previous receipt the signature value must be set to "0" – number zero	HMAC_from_ previous_receipt
transDate	Date at which the transaction was performed.	YYYY-MM-DD Do NOT use time zone or combined date and time format.	2014-01-24 2015-10-29
transTime	Time at which the transaction was performed.	hh:mm:ss Use ss=00 as default value if no information of seconds are available. Same as within the SAF-T export. Do NOT use time zone or combined date and time format.	23:59:59
nr	Transaction number. This must be a unique, sequential number within a journal. This will be the same as the number stated on the issued receipt	No leading or ending spaces.	123456789
transAmntIn	The amount involved in the transaction, including VAT.	Decimal Data Type: Numerical field with two decimals. Decimal separator "." (dot). No thousand separators. No leading or ending spaces.	1250.00 -1250.00 0.00
transAmntEx	The amount involved in the transaction, excluding VAT.	Decimal Data Type: Numerical field with two decimals. Decimal separator	1000.00 -1000.00 0.00

		"." (dot). No thousand separators. No leading or ending spaces.	

2.3.4 Practical example

Principle example of signing of data with HMAC:

1 - Data to be signed:

HMAC(signature)_from_previous_receipt;2014-01-24;23:59:59;123456789;1250.00;1000.00

«signature»;« transDate»;«transTime » ;«nr»;«transAmntIn»;«transAmntEx »

2 – Mix secret key with data and generate hash:

a) Secret key XOR inner padding = 64 bit data block (ipad)
b) Concatenate data to be signed with 64 bit data block (ipad)
c) Generate hash of concatenated data

3 – Mix secret key with hash and generate HMAC (output)

a) Secret key XOR outer padding = 64 bit data block (opad)
b) Concatenate hash (step 2) with 64 bit data block (opad)
c) Generate hash of concatenated data (HMAC)

Store HMAC (signature) for each receipt in electronic journal along with other receipt data

3 Key Generation and Management

Within the Norwegian Tax Administration keys are handled with a high level of secrecy. The objective of key management is to achieve a situation in which the private key or secret key cannot be revealed or abused. Therefore, great responsibility rests with the software vendors to protect these keys.

This section presents guidelines and best practices for key generation and management.

3.1 Responsibility of software vendor

The software vendor must do a risk assessment based on the circumstances they are facing in the following:

- Protection of private/secret key used by the ECR/POS software available to their customers.
- Protection of private/secret key within the software vendor company premises.

The conclusions and actions taken must be written down and be the basis for the management of secret key(s).

The guidance below describes cryptographically secure methods of generating keys. The software vendor must document and justify their selected way of generating their keys being used.

It is not desirable to implement explicit detailed demands on key management and generation. Therefore as long as the software developer has made a risk assessment using an industry framework, and generated keys along with industry best practices they are compliant.

The consequence of private/secret keys are being compromised is that the software vendor must update their software with new key pairs and follow the process of sending keys to the Tax Authorities.

3.2 Generation

When generating the HMAC key or RSA key pair the Norwegian Tax Administration requires the following key size:

- 1024-bit for RSA
- 128-bit for HMAC

For the generation of the HMAC key and RSA key pair a cryptographically secure PRNG (pseudorandom number generator) should be used. The system vendor should make sure that the PRNG is a well-known algorithm and well seeded, and should perform self-tests during startup. Having a malfunctioning or incorrectly seeded PRNG is one of the most common reasons for vulnerable keys.

Some further guidelines:

- The HMAC key should simply consist of random bits. Direct use the output of the random number generator can be used.

- The RSA key pair generator requires a random input to find the random primes of half the key size. This generator will likely use a high number of random bits.
- It is strongly recommended to only choose 8-bit multiples for the RSA key size in order to reduce compatibility issues.

RSA-specific notes:

- It is best to choose a public exponent in advance, 0x010001 (65537). For example the fourth prime of Fermat.
- RSA private keys should contain CRT parameters so as to make it straightforward to speed up RSA private key operations.

3.3 Distribution

The generated keys shall be transported (when necessary) using secure channels. The distribution of the public key (asymmetric encryption using RSA) and the secret key (symmetric encryption using HMAC) to the Norwegian Tax Administration is done along with the Product declaration for the Cash Register System via the Altinn portal. Use form RF-1348 in the Altinn portal for this purpose.

Sharing of secret keys with other parties must not be done, unless stated by an industry agreement with the participation of the Norwegian Tax Administration. No such agreement is done per date of this document.

3.4 Storage

The basis of key management is to ensure that the keys are stored in a secure manner. What constitutes a secure manner depends on how the environment for each cash register system is structured.

Regardless of the environment and whether the key is stored internally or externally, the following general protective measures should be considered:

- Developers must understand where cryptographic keys are stored within the application. Understand what memory devices the keys are stored on.
- Limit the amount of time the key is held in plaintext form, for example in volatile memory.
- Keys should never be stored in plaintext format, and humans prevented from viewing it in plaintext.
- Keys should be protected on both volatile and persistent memory, ideally processed within secure cryptographic modules.
- Keys should be stored so that no other than the privileged persons get access to it in plaintext form.

3.5 Accountability

Accountability involves the identification of those that have access to, or control of, cryptographic keys throughout their lifecycles. This can be an effective tool to help prevent key compromises and to reduce the impact of compromises once they are detected. Although it is preferred that no

humans are able to view keys, as a minimum, the key management system should account for all individuals who are able to view plaintext cryptographic keys.

In addition, more sophisticated key-management systems may account for all individuals authorized to access or control any cryptographic keys, whether in plaintext or cipher text form.

3.6 Compromising of key to third party

If a key is compromised, the affected product declaration must be revoked using the Altinn portal and form RF-1348 "Systemleverandørers produkterklæring av kassasystemer". The reason code selected for revoking must be "endring av privatnøkkel". This will trigger an alert to inform relevant personell at Skatteetaten. As with all other systems with revoked products declarations, affected customers must also be informed that they no longer are using an approved system.

4 Resources

NIST SP 800-133 Recommendation for Cryptographic Key Generation
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133.pdf

NIST 800-57 Recommendation for Key Management – Part 1: General (Revision 3)
http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

RFC-4107 Guidelines for Cryptographic Key Management
https://tools.ietf.org/html/rfc4107

RFC-2104 HMAC: Keyed-Hashing for Message Authentication
https://www.ietf.org/rfc/rfc2104.txt