

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 26, 2018

Y. Dong
L. Xia
Huawei
May 25, 2018

The Data Model of Network Infrastructure Device Infrastructure Layer
Security Baseline
draft-dong-sacm-nid-infra-security-baseline-01

Abstract

This document is one of the companion documents which describes the infrastructure layer security baseline YANG output for network infrastructure devices. The infrastructure layer security baseline covers the security functions to secure the device itself, and the fundamental security capabilities provided by the device to the upper layer applications. In this specific document, the integrity measurement, cryptography algorithms, key management, and certificate management are sorted out to generate the data model.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 26, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Infrastructure layer security baseline	3
2.	Terminology	4
2.1.	Key Words	4
2.2.	Definition of Terms	4
3.	Tree Diagrams	4
4.	Data Model Structure	5
4.1.	Integrity measurement	5
4.2.	Cryptography security	6
4.2.1.	Symmetrical cryptography	7
4.2.2.	Asymmetrical cryptography	8
4.2.3.	Hash function	10
4.2.4.	Message authentication code	10
4.2.5.	Key derivation function	11
4.3.	Key management	11
4.3.1.	Key generation	12
4.3.2.	Key distribution	13
4.3.3.	Key store	13
4.3.4.	Key update	13
4.3.5.	Key backup	14
4.3.6.	Key destroy	14
4.4.	Cert management	14
4.4.1.	Cert management	15
4.4.2.	CRL management	16
5.	Infrastructure Layer YANG Module	17
6.	IANA Considerations	26
7.	Security Considerations	26
8.	Acknowledgements	26
9.	References	26
9.1.	Normative References	26
9.2.	Informative References	26
	Authors' Addresses	27

[1.](#) Introduction

Network devices such as switches, routers, and firewalls are the

fundamental elements that a network is composed of. The vulnerabilities of a network device are always exploited by attackers to start up eavesdropping, spoofing, and man-in-middle attacks etc. Hence it is significant to assess the security postures for identifying the possible threats and vulnerabilities of a network

device in anytime. The SACM working group is aim to provide such a mechanism to acquire the posture information, which including the security related configuration and status attributes, on the target devices and evaluate their security postures by comparing with the pre-defined benchmarking criteria. Furthermore, the evaluation results are able to be the guidance to enforce the corresponding security hardening measurement on the devices under assessment. But this hardening process is out of scope of this draft.

This draft and each of the companion document define a subset of posture information that have to be collected for the assessment purpose mentioned above. This entire set of posture information is so called security baseline of a network device that is proposed in the companion draft [I-D.[draft-xia-sacm-dp-security-profile](#)]. The proposed security baseline is presented in the fashion of yang data model. And the security baseline yang data model can be requested or subscribed by a collector agent such as a yang push client [[draft-birkholz-sacm-yang-content](#)]. The output of such a collector agent is then encapsulated into the SACM content and statement elements [[draft-ietf-sacm-information-mdoel](#)] and published to other SACM components (e.g. repository and evaluator) [[draft-mandm-sacm-architecture-01](#)]. Please note that document is only focus on the yang data model of security baseline, the messaging mechanisms is out of scope of this document. They are specified in other documents.

1.1. Infrastructure layer security baseline

In general, the entire security baseline of a network device is divided into three layers, namely the application layer, the network layer, and the infrastructure layer. This document focus on the data model on infrastructure layer. The infrastructure layer security baseline herein refers to the configuration and status attributes of security functions that secure the device itself, and the fundamental security capabilities provided by the device to the upper layer applications. More specifically, the essential configurable and key status attributes of the following function/capability modules are

sorted out to generate the infrastructure layer security baseline data model.

- o Integrity measurement: the integrity measurement herein refers to the functions such as trust computing to protect the device against the replacement and/or tampering attacks. For example, the trust boot and/or secure boot provide the integrity validation service for the kernel and early stage executable code (bios and bootloader) in bootstrapping phases, and the digital signature protect the upper layer software applications against the tampering attacks in software updating phases.

- o Cryptography algorithms: the cryptographic algorithms are the most important capabilities that the device provides to the upper layer security applications. For example, the symmetric (e.g. DES, AES) and asymmetric (eg. RSA, ECC) cryptographic algorithms can be used for sensitive data encryption, and peers authentication. And the key derivation function (KDF) can be used for secret key generation and passcode storage.
- o Key management: the cryptographic key (pair) and its associated algorithm provide various security features for network devices. How we manage the key (pair) provisioned in a network device is a critical issue. The key management covers the attributes to show how the key (pair) is managed in the key's lifecycle (e.g. from generation to destroy).
- o Certificate management: the certificates are normally provided by the device for authentication purpose. The certificate management refers to how the certificates and the certificates revocation list (CRL) is requested, updated, and validated in the device.

The practical security baseline of a network device depends on the device type, the supported features, the requirements of operators and enterprises, and the role it plays exactly in the network. Owing to such a number of variance, it is impossible to design a comprehensive and unified data model for all devices. Thus the proposed data model in this document is only used to benchmark the most widely deployed security related functions and capabilities. And we would like it to be an extensible model so that more attributes are able to be added as per the practical use case

scenario.

[2.](#) Terminology

[2.1.](#) Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

[2.2.](#) Definition of Terms

This document uses the terms defined in[I-D.ietf-sacm-terminology].

[3.](#) Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

Dong & Xia

Expires November 26, 2018

[Page 4]

Internet-Draft Network Device Infra. Layer Sec. Baseline

May 2018

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

[4.](#) Data Model Structure

As mentioned above, the top-level structure of the data model is shown in the following figure. There are four subtrees in the tree diagram. Each of the following sub-sections specifies the detail of an individual subtree.

module: infrastructure-layer-baseline

```

+--rw infrastructure-layer-baseline
+--rw integrity-measurement
|   . . . . .
+--rw cryptography-algorithms
|   . . . . .
+--rw key-management
|   . . . . .
+--rw certificate-management
    . . . . .

```

[4.1.](#) Integrity measurement

The purpose of integrity measurement is to prevent the upper layer software applications, kernel, and early stage executable code (e.g. BIOS and bootloader) from replacement and/or tampering in bootstrapping and updating phases. Trusted boot and secure boot are the two widely used techniques for protecting the device bootstrapping. The read-only root of trust (RoT) should be always stored in a SoC or TPM chip. For software updating, digital signature has been demonstrated as a powerful tool to provide the integrity protection service. In using digital signature, the employed hash function and signature algorithm must be strong enough so that attackers cannot force crack them in a short period of time. Moreover, the public key used for verifying the signature should be stored properly. For example, it can be wrapped in a certificate of the software vendor or stored in the read-only SoC or TPM.

```

module: integrity-measurement
+--rw integrity-measurement
+--rw bootstrapping
|   +--rw trust-boot
|   |   +--ro tmp-version          string
|   |   +--rw tpm-enable          boolean
|   |   +---u hash-function
|   |   +--rw pcr-record* [pcr-number]
|   |       +--ro pcr-number      uint8
|   |       +--ro measurement-item enumeration
|   |       +--ro pcr-value       string
|   |       +--ro pcr-benchmark-value string
|   |       +--ro verify-result   boolean
|   +--rw secure-boot
|       +--ro soc-model          string

```

```

|      +--ro measurement-item*          enumeration
|      +---u hash-function
|      +---u signature-algorithm
|      +--ro verification-public-key
|          +--ro key-name                string
|          +--ro key-length              unit16
|          +--ro key-store-medium        enumeration
+--rw software-update
    +---u hash-function
    +---u signature-algorithm
    +--ro verification-public-key
        +--ro key-name                string
        +--ro key-length              unit16
        +--ro key-store-medium        enumeration

```

4.2. Cryptography security

Almost all the security features of communication network are built on the basis of modern cryptography. For example, the cryptographic algorithms are usually used to perform transmission data encryption and peers authentication. However, as the computing capability of the present computing system is getting faster and faster, more and more cryptographic algorithms can be brute force cracked in a short period of time. Therefore the algorithm has to be selected appropriately for different use case scenarios. And the configuration parameters must be set within an appropriate range so that the used algorithm is strong enough.

As a fundamental capabilities provided by the device, the practical configurations of each supported cryptographic algorithm varies as per the upper layer application that employs the algorithm. This section organizes the algorithms and their configuration parameters

into groupings so that the upper layer applications can reference/reuse them appropriately.

In general, this section covers the following cryptographic algorithm groupings:

- o Symmetric algorithms and their configurable parameters.

- o Asymmetric algorithms and their configurable parameters.
- o Hash functions.
- o Message authentication code (MAC) methods and their configurable parameters.
- o Key derivation functions (KDF) and their configurable parameters.

All the groupings enable the collection of the specific algorithms and their parameters on a case-by-case basis.

[4.2.1.](#) Symmetrical cryptography

The symmetric algorithms are typically used for providing data confidential service. The encryption and decryption process of symmetrical algorithms make use of two identical keys. And, most of the symmetrical algorithms are typically belong to either block ciphers or stream ciphers.

Block cipher: block cipher divides the plaintext in to a number of blocks with a constant bit length. And the last plaintext block should be filled to fit the bit length requirement. Then each of the plaintext blocks is encrypted individually. However, if a plaintext piece repeats several times in a long data stream, it is easier for an attacker to guess the original plaintext from the repeated ciphertext. Hence, some other operation modes of block cipher, including cipher block chaining (CBC) mode, cipher feedback (CFB) mode, counter mode (CRT), and Galois counter mode (GCM), are proposed to introduce a random bit stream, which is named initialization vector (IV), to augment the randomness of the original plaintext. The used random number generator must meet the randomness requirement so that the IV value is unpredicted. In addition, the bit length of IV should be the same as the bit length of a plaintext block for most block cipher working mode. But for CRT and GCM, the length of IV is optional.

Stream cipher: unlike block cipher, which encrypt a single plaintext block at one time, stream-cipher encrypt every bit of a plaintext

separately. The stream cipher algorithms also use IV to increase the

randomness of the original plaintext.

```
grouping: symmetric-cryptosystem
  +--rw (algorithm-type)
    +--:(stream-cipher)
      | +--rw algorithm          identityref
      | +--rw iv-length         unit16
      | +--rw iv-randomness     decimal64
    +--:(block-cipher)
      +--rw algorithm          identityref
      +--rw operation-mode     identityref
      +--rw padding-method     identityref
      +--rw iv-length         unit16
      +--rw iv-randomness     decimal64
```

[4.2.2.](#) Asymmetrical cryptography

The asymmetric cryptography is also called public key cryptography. In contrast to the symmetric one, asymmetric cryptography always employs a key pair that contains two different keys to deal with the encryption and decryption work. The private key in the key pairs is held and used only by the owner. The other key in the key pairs is theoretically public to everyone. The asymmetric cryptography algorithms are not only able to provide data encryption, but also provide authentication and/or integrity protection services (e.g. digital signature).

Asymmetric encryption: RSA is the most commonly used asymmetrical encryption algorithm. In the use of RSA, the smaller the public exponent is, the higher efficiency the algorithm has. In the other side, it will be much easier to crack the algorithm and recover the original plaintext if the public exponent is too small. Hence it has to trade off the value of public exponent. In addition, the RSA is recommend to use optimal asymmetrical encryption padding (OAEP) to fill up the original plaintext.

```
grouping: encryption-algorithm
  +--rw encryption-algorithm
    +--rw rsa-attributes
      +--rw algorithm          identityref
      +--rw padding-method     identityref
      +--rw public-key
        +--rw public-exponent  unit32
        +--rw modulo-value     unit32
```

Digital signature: digital signature is a powerful tool to provide integrity protection. DSA, RSA, and ECDSA are three of the most

popular signature algorithms. By using RSA in digital signature, it is better to use PSS for padding. If the data is required to be encrypted and signed at the same time, it is suggest to sign the data before encrypting.

```

grouping: signature-algorithms
  +--rw (asymmetric-algorithms)
    +--:(rsa)
      |   +--rw algorithm                identityref
      |   +--rw padding-method          identityref
      |   +--rw public-key
      |     +--rw public-exponent        unit32
      |     +--rw modulo-value           unit32
    +--:(dsa)
      |   +--rw temporary-key
      |     |   +--rw key-length          unit16
      |     |   +--rw randomness          decimal64
      |   +--rw prime-number
      |     +--rw prime-modulo            unit32
      |     +--rw prime-order             unit32
    +--:(ecdsa)
      +--rw temporary-key
      |   +--rw key-length                unit16
      |   +--rw randomness                decimal64
      +---u hash-function
      +--rw prime-modulo                  unit32
      +--rw prime-order                   unit32
      +--rw ec-parameters
        +--rw coefficient-a               unit16
        +--rw coefficient-b               unit16

```

Key exchange: key exchange is meant to establish key pairs between communication peers. The peers send key material rather than key itself to each other.

```
grouping: key-exchange
  +--rw (key-exchange)
    +--:(dh)
      | +--rw dh-handshake
      |   +--rw prime-number-length    unit32
      |   +--rw public-integer-length  unit32
    +--:(ecdh)
      +--rw ecdh-handshake
        +--rw prime-modulo              unit32
        +--rw ec-parameters
          | +--rw coefficient-a          unit16
          | +--rw coefficient-b          unit16
        +--rw primitive-elements
          +--rw coordinate-x             unit16
          +--rw coordinate-y             unit16
```

[4.2.3.](#) Hash function

Hash functions are normally used to perform integrity measurement. The output of a Hash function is a digest with a constant bit length for a segment of messages or code. The digest is unique and unable to be reconstructed if the original message/code is tampered. The Hash function is widely used in digital signature, message authentication code, password hash storage, and etc.

```
grouping: hash-function
  +--rw hash-function
    +--rw algorithm          identityref
    +--rw padding-method     identityref
    +--ro digest-length      unit16
```

[4.2.4.](#) Message authentication code

Similar to digital signature, message authentication code (MAC) is another method to provide integrity protection service. MAC applies hash function or block cipher algorithms on the message plaintext coupled with a pre-shared session key. It must be noted that, it is unsafe if simply extend the message with the session key.

```

grouping: message-authentication-code
  +--rw (message-authentication-code)
    +--: (hmac)
      | +--rw message-structure      enumeration
      | |   {prefix|postfix|hmac structure}
      | +---u hash-function
      | +--rw session-key
      |   +--rw key-length            unit16
      |   +--rw randomness            decimal64
    +--: (cmac)
      +--rw block-cipher-algorithm    identityref
      +--rw block-length               unit16
      +--rw iv-length                  unit16
      +--rw randomness                 decimal64

```

[4.2.5.](#) Key derivation function

Key derivation function derives one or more keys from a master key or entered password. A salt value is generated by a random number generator to introduce the randomness of the derived keys.

```

grouping: key-derivation-function
  +--rw (algorithm)
    +--: (pbkdf2)
      | +---u hash-function
      | +--rw iteration                unit16
      | +--rw derived-key-length       unit16
      | +--rw code-length              unit16
      | +--rw salt-attributes
      |   +--rw salt-length            unit16
      |   +--rw randomness             decimal64
    +--: (scrypt)

```

+-rw code-length	unit16
+-rw cpu-memory-usage	unit16
+-rw block-size	unit8
+-rw parallelization	unit8
+-rw derived-key-length	unit16
+-rw salt-attributes	
+-rw salt-length	unit16
+-rw randomness	decimal64

[4.3.](#) Key management

Cryptographic key plays the most important role in a cryptographic system. . If the key is disclosed or tampered, the corresponding service is not reliable any more. Hence the network device must provide the confidentiality and integrity protection for a key in its entire lifecycle. This section contains a list of key (pair) and

their configuration/status parameters corresponding to different lifecycle phases. Each of the key (pair) is used in a specific use case.

```

module: key-management
  +-rw key-management* [key-name]
    +-rw key-name          string
    +-rw key-length*       unit16
    +-rw lifetime          unit32
    +-rw key-type          enumeration
    +-rw num-of-keys       unit8
    +-rw key-generation
    | . . . . .
    +-rw key-distribution
    | . . . . .
    +-rw key-store
    | . . . . .
    +-rw key-backup
    | . . . . .
    +-rw key-update
    | . . . . .
    +-rw key-destroy
    . . . . .

```

[4.3.1.](#) Key generation

There are three types of commonly used key generation methods. The first method is on the basis of random number generator. In this method, the referenced random number generator has to ensure the generated key is unpredicted. The second key generation method is based on the manual entered password. However, the entered password is not meet the randomness requirement. In this case, a key derivation function (e.g. PBKDF2) is applied to derive the key. The last key generation method is key exchange such as Diffie-Hellman (DH) protocol. This kind of method requires the peers to authenticate each other before exchange the key material.

```

submodule: key-generation
  +---rw key-generation
    +---: (random-number-generator)
      | +---rw key-randomness          decimal64
    +---: (key-derivation-function)
      | +---u key-derivation-function
    +---: (key-exchange)
      +---rw cert-name                string
      +---u key-exchange

```

[4.3.2.](#) Key distribution

Key distribution aims to send the generated keys to authorized entities in a secure fashion. The confidentiality and integrity issues of the key in distribution are usually addressed by using either a secure transport protocol or digital envelop.

[[I-D.ietf-netconf-tls-client-server](#)], IPsec [[I-D.draft-tran-ipsecme-yang](#)], or SSH [[I-D.ietf-netconf-ssh-client-server](#)], or digital envelop.

```

submodule: key-distribution
  +---rw key-distribution?
    +---rw symmetrical-key
      +---: (secure-transport-protocol)
        | +---rw tls-config
        | |   [I-D.ietf-netconf-tls-client-server]
        | +---rw ipsec-config
        | |   [I-D.draft-tran-ipsecme-yang]

```

```

|   +---rw ssh-config
|       [I-D.ietf-netconf-ssh-client-server]
+---: (digital-envolop)
      +---u symmetric-algorithm
      +---rw encryption-key-name      string
      +---rw encryption-key-length    unit16

```

[4.3.3.](#) Key store

A typical key management system has three layers. The master keys that consumed by upper layer applications are in the top layer. The key in the middle layer, which is called key encryption key (KEK), is used to encrypt the master keys. And the KEK itself is encrypted by the root key which stays in the bottom layer of the three layer key management system.

```

submodule: key-store
  +---rw key-store
    +---ro store-medium {TPM|HSM|HDD}      enumeration
    +---rw key-component* [component-name]
      +---rw component-name                string
      +---ro store-medium                  enumeration

```

[4.3.4.](#) Key update

Network device must update the key in a reasonable period of time. Otherwise the long term used key will attract attackers to crack it. The practical update period of a certain key depends on the application the key serves and the strength (i.e. bit length) of the key itself.

```

submodule: key-update
  +---rw key-update
    +---rw next-update-time      yang-type:date-and-time
    +---rw hold-expired-key      boolean
    +---rw update-mode
      +---: (manual)
      |   +---rw manual-enable    boolean
      +---: (auto)
          +---rw auto-enble       boolean
          +---rw update-period    unit8

```

[4.3.5.](#) Key backup

The loss of keys will lead to data loss. Therefore, according to the different use case scenarios, a key (pair) may need to backup. It is better to divide the key into several parts and store them into different storage devices.

```
submodule: key-backup
  +--rw key-backup?
    +--rw backup-enable          boolean
    +--rw backup-expire-time     yang-type:date-and-time
    +--rw backup-component* [component-name]
      +--rw component-name      string
      +--ro backup-medium       enumeration
```

[4.3.6.](#) Key destroy

The key and its associated key material must be destroyed when it is expired. Otherwise the expired key will be used by attackers to decrypt the data encrypted by this key. Also, the expired key can be used to analysis the cryptosystem.

```
submodule: key-destory
  +--rw key-destory
    +--rw method      {one|zerod|random number} enumeration
    +--rw number-of-times      unit8
```

[4.4.](#) Cert management

The TLS/DTLS and IPsec have been demonstrated as powerful security tools to provide data confidentiality and integrity services between network elements. In order to protect the TLS/DTLS or the IPsec connection against man-in-middle attack, peers have to authenticate from each other before connection establishing. The pre-shared key and the certificate are two of the most widely used methods to authenticate peers' identities. However, it requires to re-configure

the pre-shared keys on all other endpoints/network elements if an additional network device is added in network. This complicated re-configuration process is easy to make errors. In the other hand, certificate is an idea way to extend authentications to a much larger

scale of network. Peers request certificates that contain their entity information and public keys from certification authority (CA) in advance. The connection will be established only if the certificates are verified.

For a specific network device, such as switch and router, the certification service normally includes certificates request and updating, certificates validity check.

```
module: cert-management
  +--rw cert-management
    +--rw cert-management
      | . . . . .
    +--rw crl-management
      . . . . .
```

[4.4.1.](#) Cert management

A cert request file that contains the device public key and entity information is sent to the CA to apply a certificate. A CMP session is configured to request and update the certificates. A build-in default certificate in the device is used for identity authentication for CMP session. And the certificate must be updated in a reasonable period of time via CMP session.

```

module: cert-management
  +--rw cert-management* [cert-name]
    +--rw cert-name          string
    +--ro version            string
    +--ro serial-number      string
    +--ro signature-algorithm identityref
    +--ro issuer-name        string
    +--rw cert-request
      | +--rw cmp-session-name string
    +--ro validity
      | +--ro start-time      yang-type:date-and-time
      | +--ro end-time        yang-type:date-and-time
    +--ro subject-public-key
      | +--ro public-key-algorithm identityref
      | +--ro public-key-length unit16
      | +--ro exponent        unit32
    +--rw cert-auto-update
      +--rw cert-name          string
      +--rw pki-domain-name    string
      +--rw cmp-session-name   string
      +--rw auto-update-enable boolean
      +--rw trigger-condition
        +--rw validity-percentage-number unit8

grouping: cmp-session-config
  +--rw cmp-session-config* [session-name]
    +--rw domain-name          string
    +--rw session-name         string
    +--rw entity-name          string
    +--rw key-name              string
    +--rw ca-server-name       string
    +--rw default-cert-name     string
    +--rw cmp-server-url       string

```

[4.4.2.](#) CRL management

The certificate revocation list (CRL) contains the invalid/expired certificates. It is equivalent to a blacklist of certificates issued by CA. The validity of a received cert is able to be checked by comparing with the CRL. The CRL need to update from CA by either an automatic or manual way.

```
submodule: srl-management
  +--rw srl-management
  +--rw cert-validity-check-enable      boolean
  +--rw srl-update
    +--rw previous-update-time          yang-type:date-and-time
    +--rw auto-update
      | +--rw auto-update-enable        boolean
      | +--rw update-period              unit32
      | +--rw next-update-time          yang-type:date-and-time
      | +--rw update-method {http|ldap} enumeration
    +--rw manual-update
      +--rw manual-update-enable        boolean
      +--rw update-method {http|ldap} enumeration
```

5. Infrastructure Layer YANG Module

This section shows a fraction of the infrastructure layer security baseline YANG modules.

```
module ietf-integrity-measurement{
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-integrity-measurement";
  prefix "im";

  import ietf-yang-types{
    prefix yang;
    reference
      "RFC6991: Common Yang Data Types";
  }

  organization
    "Huawei Technologies";

  contact
    "Yue Dong: dongyue6@huawei.com"
    "Liang Xia: Frank.xialiang@huawei.com"

  description
    "This module defines the configuration and status parameters of the
```

functions that provide the integrity services in the bootstrapping and software updating phases.";

```
identity hash-algorithms {  
  description  
    "base identities of hash algorithms options";  
}
```

```
identity md5 {  
  base hash-algorithms;  
  description  
    "The MD5 algorithm";  
}
```

```
identity sha1 {  
  base hash-algorithms;  
  description  
    "The SHA-1 algorithm";  
  reference  
    "RFC3174: US Secure Hash Algorithm 1 (SHA1).";  
}
```

```
identity sha224 {  
  base hash-algorithms;  
  description  
    "The SHA-224 algorithm.";  
  reference  
    "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and  
    HKDF).";  
}
```

```
identity sha256 {  
  base hash-algorithms;  
  description  
    "The SHA-256 algorithm.";  
  reference  
    "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and  
    HKDF).";  
}
```

```
identity sha384 {
```

```

    base hash-algorithms;
    description
        "The SHA-384 algorithm.";
    reference
        "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and
        HKDF).";
}

```

```

identity sha512 {
    base hash-algorithm;
    description
        "The SHA-512 algorithm.";
    reference
        "RFC6234: US Secure Hash Algorithms (SHA and SHA based HMAC and
        HKDF).";
}

```

```

}

identity rsa-algorithms {
    description
        "rsa algorithms with different key length";
}

identity rsa1024 {
    base rsa-algorithms;
    description
        "The RSA algorithm using a 1024 bit key";
    reference
        "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
        Cryptography Specifications 2.1"
}

identity rsa2048 {
    base rsa-algorithms;
    description
        "The RSA algorithm using a 2048 bit key";
    reference
        "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
        Cryptography Specifications 2.1"
}

identity rsa3072 {

```

```

    base rsa-algorithms;
    description
        "The RSA algorithm using a 3072 bit key";
    reference
        "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
        Cryptography Specifications 2.1"
}

identity rsa4096 {
    base rsa-algorithms;
    description
        "The RSA algorithm using a 4096 bit key";
    reference
        "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
        Cryptography Specifications 2.1"
}

identity rsa7680 {
    base rsa-algorithms;
    description
        "The RSA algorithm using a 7680 bit key";
    reference

```

```

        "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
        Cryptography Specifications 2.1"
}

identity rsa15360 {
    base rsa-algorithms;
    description
        "The RSA algorithm using a 15360 bit key";
    reference
        "RFC3447: Public-Key Cryptography Standards (PKCS) #1: RSA
        Cryptography Specifications 2.1"
}

identity rsa-padding {
    description
        "The identities of padding methods for rsa.";
}

identity oaep {

```

```

    base rsa-padding;
    description
        "The OAEP padding method for RSA.";
}

identity pss {
    base rsa-padding;
    description
        "The PSS padding method for RSA.";
}

container integrity-measurement {
    container bootstrapping {
        container trust-boot {
            leaf tpm-version {
                type string;
                description
                    "version of the tpm chip";
            }

            leaf tpm-enable {
                type boolean;
                description
                    "switch of the trust boot function";
            }

            uses hash-function

            list pcr-record {

```

```

    key "pcr-number";

    leaf pcr-number {
        type uint8;
        description
            "Number of pcr register";
    }

    leaf measurement-item{
        type enumeration {
            enum bios;
            enum bootloader;

```

```

        enum kernel;
        enum patch;
    }
    description
        "This property shows which item is measured and recored by
        the pcr";
}

leaf pcr-value {
    type string;
    description
        "The practical measurement value";
}

leaf pcr-benchmark-value {
    type string;
    description
        "The pre-defined benchmark criterion";
}

leaf verify-result {
    type boolean;
    description
        "The benchmark result for each pcr recorded value";
}
}
}

container secure-boot {
    leaf soc-model {
        type string;
        description
            "Model of the used SoC";
    }

    leaf-list measurement-items {

```

```

type enumeration {
    enum bios;
    enum bootloader;
    enum kernel;
    enum patch;

```



```

    }
    description
        "List of the items to be measured in the secure boot
        process";
}

uses hash-function

uses signature-algorithm

container verification-pub-key {
    leaf key-name {
        type string;
        description
            "Name of the public key for verification";
    }

    leaf key-length {
        type unit16;
        description
            "Length of the public key"
    }

    leaf store-medium {
        type enumeration {
            enum tmp;
            enum soc;
            enum hdd;
            enum hsm;
        }
        description
            "This property describes where the public key stores"
    }
}

}

}

container software-update {
    uses hash-function;

    uses signature-algorithm;

    container verification-pub-key {

```

```

    leaf key-name {
        type string;
        description
            "Name of the public key for verification";
    }

    leaf key-length {
        type unit16;
        description
            "Length of the public key";
    }

    leaf store-medium {
        type enumeration {
            enum tpm;
            enum soc;
            enum hdd;
            enum hsm;
        }
        description
            "This property describes where the pub key stores"
    }
}
}
}

```

```

grouping hash-function {
    description
        "A group of Hash functions and their parameters";

```

```

    leaf algorithm {
        type identityref {
            base "hash-algorithm";
        }
        description
            "Identities of the used Hash algorithm";
    }

```

```

    leaf padding-method {
        type identityref;
        description
            ""
    }

```

```

    leaf digest-length {
        type unit16;
        description
            "The length of the Hash output";
    }

```

Internet-Draft Network Device Infra. Layer Sec. Baseline

May 2018

```
}  
}
```

```
grouping signature-algorithms {  
  "A group of algorithms and their configurable parameters for digital  
  signature";
```

```
  choice algorithm-type {  
    case rsa {  
      leaf algorithm {  
        type identityref {  
          base "rsa-algorithm";  
        }  
        description  
          "identities of the rsa algorithms for digital signature";  
      }  
  
      leaf padding-method {  
        type identityref;  
        description  
          "identities of padding method for the used algorithm"  
      }  
  
      container pub-key {  
        leaf public-exponent {  
          type unit32;  
          description  
            "value of public exponent";  
        }  
  
        leaf modulo-value {  
          type unit32;  
          description  
            "value of modulo";  
        }  
      }  
    }  
  }
```

```
  case dsa {  
    container temporary-key {  
      leaf key-length {  
        type unit16;  
      }  
    }  
  }
```

```

        description
            "The length of the temporary key.";
    }

```

```

    leaf randomness {
        type decimal64;
    }

```

```

        description
            "This value represents the randomness of this key.";
    }
}

container prime-number {
    leaf prime-modulo {
        type unit32;
        description
            "value of modulo";
    }

    leaf prime-order {
        type unit32;
        description
            "value of prime number";
    }
}

case ecdsa {
    container temporary-key {
        leaf key-length {
            type unit16;
            description
                "The length of the temporary key that is generated by a
                random number generator.";
        }

        leaf randomness {
            type decimal64;
            description
                "This value represents the randomness of the key. It is
                generated by a tool like sts 2.1.";
        }
    }
}

```


8. Acknowledgements

TBD

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

9.2. Informative References

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", [draft-ietf-netconf-ssh-client-server-05](#) (work in progress), October 2017.

Dong & Xia

Expires November 26, 2018

[Page 26]

Internet-Draft Network Device Infra. Layer Sec. Baseline

May 2018

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", [draft-ietf-netconf-tls-client-server-05](#) (work in progress), October 2017.

[I-D.ietf-sacm-information-model]
Waltermire, D., Watson, K., Kahn, C., Lorenzin, L., Cokus, M., Haynes, D., and H. Birkholz, "SACM Information Model", [draft-ietf-sacm-information-model-10](#) (work in progress), April 2017.

[I-D.ietf-sacm-terminology]
Birkholz, H., Lu, J., Strassner, J., Cam-Winget, N., and A. Montville, "Security Automation and Continuous Monitoring (SACM) Terminology", [draft-ietf-sacm-terminology-14](#) (work in progress), December 2017.

[I-D.mandm-sacm-architecture]
Montville, A. and B. Munyan, "Security Automation and Continuous Monitoring (SACM) Architecture", [draft-mandm-sacm-architecture-01](#) (work in progress), March 2018.

[I-D.tran-ipsecme-yang]

Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", [draft-tran-ipsecme-yang-00](#) (work in progress), October 2015.

[I-D.xia-sacm-nid-dp-security-baseline]

Xia, L. and G. Zheng, "The Data Model of Network Infrastructure Device Data Plane Security Baseline", [draft-xia-sacm-nid-dp-security-baseline-01](#) (work in progress), January 2018.

Authors' Addresses

Yue Dong
Huawei

Email: dongyue6@huawei.com

Liang Xia
Huawei

Email: frank.xialiang@huawei.com