PROTECTED
関係者外秘

Trusted Update Package Format Design

System Software Development Department,

InfoTech,

Connected Advanced Development Division,

Toyota Motor Corporation

Contact : Okino, Naoto (naoto.okino@toyota-tokyo.tech)

Revision History

Version	Date	Description
1.0	28/9/2021	Initial release

Purpose of this document

The purpose of this document is understanding the design of the Trusted Update Package (TUP) file format and its rationale.

Objectives of TUP format

There are the following objectives to be achieved in the TUP file format.

- Operation Guarantee
- Damage and Tamper Resistance
- Data Leakage Prevention
- Random Accessibility
- Processing efficiency of generating file
- Optimization for the Target Device

Measures

This section describes the following measures and how each contributes to achieve the objectives above.

- Single File With Proprietary Format
- TUP File Structure
- TLV (Type-Length-Value) Format
- ICV tree / ICV array
- Security

Single File With Proprietary Format

Data for update should be treated as a single file that can be configured for various purposes, rather than as "multiple files distributed with different formats defined for different purposes".

This measure contributes to the achievement of the following Objectives.

[Objective] Operation Guarantee

- Combination problems or missing of update files wouldn't occur when distributing (If multiple files are required for an update, it is difficult to guarantee the synchronization and combination of updates for each file).
- Since data in a TUP file is given linear offset, sanity check is easy when referring to it.
- Update history is easy to manage, since "update of a specific version" matches a certain file.
- As a proprietary format, it supports only files created in a specific generation environment, thereby reducing the diversity of files to be handled. If a zip file generated by any archiver would be accepted, it is difficult to guarantee compatibility and the interpretation process is easily attacked.

TUP File Structure

The following logical areas are defined in a TUP file.

- FixedHeader (FH)
- VariableHeader (VH)
- Inner Package Data
- Inner Package Metadata
- VariableFooter (VF)
- FixedFooter (FF)

FixedHeader (FH)

Fixed size header which is located at the first part of a file.

The size of FH includes the padding for the future expansion. Data in this area can be accessed with low cost due to its location. However, the size of FH should be bare minimum since it will be processed on every target which can handle TUP files.

The TUP format version and the number of the Inner Packages are stored in the first part of FH. The default endianness of the TUP file is determined by its format version. Therefore, as specification, there is no conflict with the byte-swapped value of a valid version (Valid versions shall be determined in advance and the endianness of the version value is the TUP file's endianness).

Following the number of Inner Packages, tuples of 'Type, Offset, Size' of each logical area (=VH, VF, FF) are stored. Since interpretation of each area depends on its 'Type', version and the structure of FH won't have to be changed in order to add functionality in the future.

This structure contributes to the achievement of the following Objectives.

[Objective] Operation Guarantee

- With a fixed size header, any TUP file can be handled safely even on a target with an old version.
- Future expansion of data is possible without exceeding the limits of the system as long as the padding size.

[Objective] Optimization for the Target Device

Variable sized information is stored in VH or VF and FH only contains information about these areas. Due to this policy, size and complexity of FH don't increase depending on the number of Inner Packages.

[Objective] Damage and Tamper Resistance

- Validity of FH can be guaranteed by making it the target of an ICV array or ICV tree. Though TUP files can be structured in a way its FH is not the target of the checking, such a file should not be publicly available.
- Since FH is the first thing to interpret, encryption information of FH itself cannot be contained in a TUP file. In a case where FH needs to be encrypted, the encryption/decryption method should be agreed separately when delivery so that the interpretation system can decrypt the FH.

VariableHeader (VH)

VH can be one of the following formats.

- Simple format which contains an array of each Inner Package's offset and size.
    - VH only contains information of areas of the number of the Inner Packages.
- Extensible format which consists of a collection of TLV format records.
    - VH can contain any metadata. In case a certain data allocation is necessary, which cannot be represented by the simple format, the allocation would be feasible by using this format (although the processing cost would increase) .

This structure contributes to the achievement of the following Objectives.

[Objective] Processing efficiency of generating file

The size of VH stored in FH must not match the actual data size. Therefore, developers can adopt a package file generation strategy where they start generating a TUP file when the maximum number of Inner Packages is presumed, and allow unused parts to be left in VH. In addition, when generating various types of TUP files containing common contents, the generation process can be reduced by aligning the location of parts to be shared.

[Objective] Random Accessibility

By using VH, a specific Inner Package data can be extracted with low cost. Without loading the whole TUP file, bare minimum data for update can be transferred from OTA Global Master to target systems with low cost.

[Objective] Optimization for the Target Device

- VH usually is expected to be located right after FH. In this case, on target with sufficient resources, processing cost can be reduced by reading, decrypting and verifying FH and VH (only first part) at once.
- On a target whose processing efficiency is enhanced under a specific condition (e.g. for hardware support), a TUP file can be aligned with the condition since VH can be started at any position.

Inner Package Data

Actual data of each Package can be located at any position following VH information, excluding the end of the TUP file. When a TUP file contains multiple Inner Packages, these actual data locations can be determined independently and doesn't need to be consecutive.

When storing Inner Package data, encryption is usually applied and compression is also applied if possible. If encryption or compression is applied, corresponding metadata will be contained in VF. In a case Inner Package data has multiple data with different purposes in it, different encryption or compression methods can be applied to each sub-area of Inner Package data and corresponding metadata to it would be also multiple.

In order to reduce the size, data which is able to be compressed will be compressed in a TUP generation environment. Compressed data and generated compressed block table (if necessary) will be stored. Random access to compressed data is possible by storing a compressed block table and using the offset before compression.

In case the data is encrypted with sufficient strength, encryption in a TUP generation environment can be skipped. However, available operation would be limited if decryption by OTA Global Master is not possible. In case the data needs to be kept secret but it is not encrypted or the strength is not sufficient, it can be encrypted in a TUP generation environment and then stored.

In addition, ICV of Inner Package data and its generated metadata would be calculated and stored separately. ICV is usually expected to be calculated with data 'encrypted after compressed '. It is possible to extend the behavior by generating additional metadata when exceptional process sequence is necessary.

This structure contributes to the achievement of the following Objectives.

[Objective] Optimization for the Target Device

- Actual data of the Inner Package can be started at any location. Therefore, it is possible to optimize the data to be aligned with the memory page when 'mmap a TUP file and pass its memory to the hardware security function'.
- In case the data is compressed when generating the TUP file with the update framework function, it can be decompressed on the target before delivering to some ECUs. Therefore, it is possible to reduce the cost of delivery to the target with compressed data and also reduce the burden of ECUs with low resources by decompressing transparently.
- In addition, the compression method is selectable, so processing efficiency can be improved by using the compression method supported by the target hardware.

[Objective] Data Leakage Prevention

Data that needs to be kept secret needs to be encrypted for distribution, but the security features available on the target varies depending on ECUs, and sufficient strength encryption is not always available. For updating an ECU which cannot decrypt the data by itself, the data can be encrypted and delivered to the OTA master, and the decrypted data can be delivered through a secure route and protected strongly.

[Objective] Random Accessibility

If Inner Packages are compressed and stored simply, the whole data has to be decompressed in order to extract a specific location data. From the point of view of transfer volume and processing cost, only the compressed data corresponding to the required data should be delivered. TUP can have a table of correspondence of the positions before and after compression (compressed block table). By using the table, the data at intermediate locations can be extracted and decrypted.

For data that doesn't need random access (data that always needs the whole) , a compressed block table can be omitted.

[Objective] Damage and Tamper Resistance

ICV should be generated for verification if each Inner Package data is encrypted using a method that is not tamper-proof (e.g. XTS) as the data could be destroyed randomly. In order to verify each Inner Package data when it arrives at the target ECU (after it is extracted from TUP file by OTA master), verification data should be prepared not for the entire TUP file but for each Inner Package.

Inner Package Metadata

Actual data of metadata (ICV, compressed block table) of Inner Package such as ICV value array and table can be allocated at any position of a TUP file and information to reach to those is stored in VF.

This structure contributes to the achievement of the following Objectives.

[Objective] Optimization for the Target Device

Depending on the expected use case, data arrangement can be selected. For example:

- Each metadata is placed adjacent to its original data.
- Metadata is placed together depending on its types.
- Certain alignment is guaranteed for specific metadata.

[Objective] Damage and Tamper Resistance

The actual data of each metadata can be verified with an ICV tree or ICV array. These ICV trees or ICV arrays also can be verified with upper ICV arrays.

[Objective] Processing efficiency of generating file

In order to reduce generation time of high cost metadata, metadata such as ICV or compressed block tables can be obtained without determining the location of actual data in the TUP file.

VariableFooter (VF)

Area to store metadata (such as data for decompression or verification of each area and metadata of TUP file itself). VF consists of TLV format records.

List of pairs of type and offset can be contained at the first part of VF as TLV, whose Type is 'index'. Even when there are many Inner Packages, there is no need to check all the existing TLVs by order because there is 'index' information of Inner Package metadata.

This structure contributes to the achievement of the following Objectives.

[Objective] Data Leakage Prevention

Decryption and decompression information of data in a TUP file is basically contained in VF as a TLV record. And, decryption information of VF itself, in case it is encrypted, is contained in FF.

[Objective] Damage and Tamper Resistance

The VF area is included in the scope of an ICV tree, either dedicated or shared with others, and is protected from tampering.

FixedFooter (FF)

Fixed size footer area which locates at the end of a TUP file. Information that can be generated only after all the other parts are structured such as :

- The top ICV array information
- VF encryption method
- Method to derive signature
- Signature

and checksum for casual damage detection are stored here.

If FF has to be encrypted, the encryption information cannot be stored in the TUP file. If the data is to be kept secret, additional information must be provided by the environment as in the case of FH.

This structure contributes to the achievement of the following Objectives.

[Objective] Damage and Tamper Resistance

- Tampering of a TUP file can be detected by verifying each ICV tree or ICV array from the top ICV array recursively. The validity of the top ICV array itself is guaranteed by storing the ICV of the top ICV array itself, and a signature to derive method in FF.

- Since signatures are generated for all the data preceding FF, they can be trusted if the signature is successfully verified. In order to prevent the TUP from being forged even if the target loses full control, the signature must be data that, unlike the ICV value, can be verified only but not can be generated on the target (there should be no way to generate it on the target). For this reason, if a particular signature derivation method is compromised, an appropriate revoke mechanism is required, separate from the TUP processing system.

- Damage detection using ICV is rigorous, but the cost is high because it requires analysis of a TUP file. For use cases where damage can be confirmed with sufficient accuracy and has no security impact, a simple checksum and its generation method information should be added to the end of the file. Since the checksum must be able to be verified without any additional information, it is not subject to encryption. For example, this can be used when investigating the process where a TUP file corruption happened. Even nodes that do not have the ability to interpret TUPs can still detect corruption.

TLV (Type-Length-Value) format

Data contained in areas in which TUP extensibility is required should have a structure of general TLV(Type-length-value) record. TLV could be nested. This means some type has its child TLV groups in the Value part. It is determined by the parent type whether the V part should be interpreted as TLV or not.

In case 'Value' part data needs to have specific alignment conditions, there can be padding areas between Length and Value.

This measure contributes to the achievement of the following Objectives.

[Objective] Operation Guarantee

Interpretation system can safely ignore unknown types when handling an update which includes optional functional extension.

As area for proprietary extension is reserved in TLV Type value range, additional Type can be defined that doesn't interfere with the standard TUP when extension for a specific use case.

[Objective] Optimization for the Target Device

Value part can be aligned with target hardware. It is not desirable to have padding areas in every TLV record in terms of space cost. However, the specification that defines padding for a specific TLV Type is also too complex to manage. Therefore, a specific bit is reserved in Type and if the bit is '1', it is interpreted as there is area for padding length and padding before Value.

[Objective] Data Leakage Prevention

Additional protection(encryption) can be provided with significant data or metadata by defining the TLV record type where the Value part is encrypted.

ICV tree / ICV array

In order to detect tampering and damage, the following are required.

- Can generate ICV from data to protect and check if it matches the expected value.
- The expected value itself can be checked using upper ICV.

TUP has two data structures, ICV tree and ICV array, to store ICVs in order to handle effectively:

- Consecutive and large areas that random access is expected.
- Non-consecutive and not so large areas (such as multiple ICV trees) .

ICV tree

Hierarchical ICV collection (ICV tree) can be obtained by the following steps.

- Split the original data into a certain size of blocks and calculate the ICV of each.
- If the given ICV collection size is bigger than the block size, repeat the process to calculate ICV of it.

As the ICV collection size can be significantly big depending on the original data,

- ICV collection is stored in any areas other than header/footer.
- Metadata of ICV collection is stored in VF or areas other than header/footer as TLV (the Type is 'ICV tree').

Therefore, metadata of ICV trees of each Inner Package might be stored in VF, the size of VF (and its verification cost) can be suppressed because ICV collection itself is not stored in VF.

ICV array

ICV array is located at any areas other than header/footer as a TLV record that Type is 'ICV array'. The TLV Value is an array of three elements and each element is :

- Start position in TUP file
- Size in TUP file
- ICV

of some area in the TUP file.

The area is expected to be either:

- TLV record (ICV tree or ICV array).
- Opaque data.

In the case of an ICV tree or ICV array, after validating the TLV, the data protected by the TLV can be verified recursively. In the case of opaque data, tampering checks can be done by checking if its ICV matches the expected value.

This data structure of ICV tree and ICV array contributes to the achievement of the following Objectives.

[Objective] Random Accessibility

It is possible to achieve tampering detection simply by having "ICV for the entire part" if random access is not taken into account. However, the whole file needs to be obtained to detect tampering in such a case and the overhead is unnecessarily big in a case where only a specific range of data is needed to be accessed.

By calculating each ICV of a certain block size and aggregating them to create a tree structure, only the blocks that contain required data and the ICV tree node that cover them are needed to be checked to verify the data consistency.

[Objective] Optimization for the Target Device

- The parameters for ICV calculation (e.g. block size) can be chosen depending on the target, and they are stored in the metadata along with the ICV. Therefore, if the target supports a specific ICV processing method, the parameters can be aligned with it. Otherwise, if the target is not able to process a large size block, the parameters can be adjusted.
- In an environment with sufficient memory, caching intermediate ICVs that are verified can improve the performance of actual access when checking the data using ICV tree.
- ICV tree or ICV array can be located at any place aligned with the allocation constraints of the target as long as these are able to be detected by an ICV array referring to these.

[Objective] Operation Guarantee

In order to achieve operation of delivering only truly necessary data between ECUs, if the size of target data is not an integer multiple of the block size, the area after the end in the final block should be 0-filled (masked). This is because each data cannot be checked independently when simply extracting the block sized data if the ICV contains unrelated subsequent data.

[Objective] Damage and Tamper Resistance

Area* to protect in TUP file should be either :

- Referred to directly by ICV array.
- Included in the target range of the ICV tree.

* All the data excluding one that cannot use ICVs in FF in case the padding parts are protected as well.

TLV record to check these target areas consists of tree structure as a whole. Therefore, an ICV array or ICV tree is either :

- The top ICV array itself.
- Referred to by upper ICV array.

By checking each TLV itself with upper TLV, each ICV or parameters to calculate it is protected from tampering.

The top ICV array cannot be verified in the framework of ICV array or ICV tree. Therefore, FF should contain verification information and a secured signature in order to guarantee the validity of the top ICV.

[Objective] Processing efficiency of generating file

Encryption and generation of ICV can be done independently for each target of ICV trees or ICV arrays, rather than all at once for the en.

Security

Hardware support is required to guarantee security. The following items should be used effectively if they exist and minimum guarantee should be provided even when there is no hardware support.

- Memory and storage which can store keys.
- Clock with time that cannot be tampered with freely.
- Encryption/Decryption and signature calculation that doesn't leak the result and keys to outside.
- Irreversible keys revocation.

'Encrypt-then-MAC' is usually expected to run in order to achieve data confidentiality (encryption) and authentication (tampering prevention) at the same time. Therefore, ICV should be defined for the data after encryption, but some hardware might only support ICV calculation of which before encryption. TUP also supports non-standard processing order as format, but the priority between performance and security strength should be considered during operation.

This measure contributes to the achievement of the following Objectives.

[Objective] Optimization for the Target Device

Since the available secure modules vary depending on target vehicle, the encryption/decryption parameters used in the TUP file are not fixed, but rather the appropriate one should be chosen. And, OTA master can process, deliver and emulate (in a way as safe as possible) instead of ECUs which cannot handle encryption safely.