可信更新包格式设计

InfoTech系统软件开发部,

丰田汽车公司互联先进开发部

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

修订历史

Version	Date	Description
1.0	28/9/2021	Initial release

本文档的目的

本文档的目的是了解可信更新包(TUP)文件格式的设计及其基本原理。

TUP格式的目的

在TUP文件格式中要实现以下目标。

- 操作保证
- 防损坏防篡改防数据泄露
- 随机访问
- 目标设备生成文件优化的处理效率

措施

本节描述以下措施，以及每种措施如何有助于实现上述目标。

- 具有专有格式TUP文件结构的单个文件
- TLV (Type-Length-Value)格式ICV树/ICV数组安全

Single File With Proprietary Format

用于更新的数据应被视为一个可以配置用于各种目的的单一文件，而不是“为不同目的定义不同格式的多个文件”。

这一措施有助于实现以下目标。

【目的】运营保障

- 分发时不会出现更新文件的组合问题或丢失(如果一次更新需要多个文件，则很难保证每个文件的更新同步和组合)。
- 由于TUP文件中的数据是线性偏移的，因此在引用它时很容易进行完整性检查。更新历史记录很容易管理，因为“特定版本的更新”与某个文件匹配
- 作为一种专有格式，它只支持在特定生成环境中创建的文件，从而减少了需要处理的文件的多样性。如果要接受任何一个归档器生成的zip文件，就很难保证兼容性,- 解释过程也容易受到攻击。

TUP文件结构

在TUP文件中定义了以下逻辑区域。

- FixedHeader (FH) Variable
- Header (VH) Inner Package
- Data Inner Package Metadat
- a VariableFooter (VF)
- FixedFooter (FF)
-

FixedHeader(FH)

固定大小的头文件，位于文件的第一部分。

FH的大小包括未来扩展的填充。由于该区域的位置，可以以较低的成本访问该区域的数据。但是，FH的大小应该是最小的，因为它将在每个可以处理TUP文件的目标上进行处理

TUP格式版本和内部包的数量存储在FH的第一部分。TUP文件的默认端序由其格式版本决定。因此，作为规范，与有效版本的字节交换值没有冲突(有效版本需要提前确定，版本值的端序为TUP文件的端序)。

在内部包的数量之后，存储每个逻辑区域(VH, VF, FF)的Type, Offset, Size的元组。由于每个区域的解释取决于它的类型，版本和FH的结构不必为了在未来增加功能而改变

这种结构有助于实现下列目标

【目标】运营保障

- 有了固定大小的头，任何TUP文件都可以安全地处理，即使是在具有旧版本的目标上
- 未来的数据扩展是可能的，不超过系统的限制，只要填充大小

[目标设备的目标优化

可变大小的信息存储在VH或VF中，FH仅包含有关这些区域的信息。由于这个策略，FH的大小和复杂性不会随着内部包的数量而增加。

【目标】伤害和抗篡改

- FH的有效性可以通过使其成为ICV数组或ICV树的目标来保证，尽管TUP文件可以以某种方式结构化，其FH不是检查的目标，这样的文件不应该公开可用。
- 由于FH是首先要解释的，所以FH本身的加密信息不能包含在TUP文件中。在需要对FH进行加密的情况下，应在传送时分别约定加密/解密方法，以便解释系统对FH进行解密。

可变头(VH)

VH可以是以下格式之一

- 包含每个Inner Package的偏移量和大小的数组的简单格式
  VH仅包含由TLV格式记录集合组成的内部包可扩展格式的数量区域的信息。
- VH可以包含任何元数据。以防有必要进行一定的数据分配。无法用简单格式表示的，使用这种格式就可以进行分配(尽管处理成本会增加)

这种结构有助于实现下列目标

目标提高生成文件的处理效率

存储在FH中的VH大小不能与实际数据大小匹配。因此，开发人员可以采用包文件生成策略，在假定内部包的最大数量时开始生成TUP文件，并允许未使用的部分留在VH中。此外，在生成包含共同内容的各种类型的TUP文件时，可以通过对齐待共享部件的位置来减少生成过程

【目标】随机可及性

通过使用VH，可以以较低的成本提取特定的Inner Package数据。在不加载整个TUP文件的情况下，可以以低成本将最少量的更新数据从OTA Global Master传输到目标系统。

【目标】目标设备的优化

- VH通常位于FH之后。这种情况下，在资源充足的目标上，可以通过同时读取、解密和验证FH和VH(仅第一部分)来降低处理成本。
- 对于在特定条件下(例如硬件支持)提高处理效率的目标，由于VH可以在任何位置启动，因此TUP文件可以与条件对齐

nner封装数据

每个包的实际数据可以位于VH信息之后的任何位置，但不包括TUP文件的末尾。当一个TUP文件包含多个内部包时，这些实际的数据位置可以独立确定，而不需要连续。

在存储Inner Package数据时，通常会进行加密，如果可能的话也会进行压缩。如果应用加密或压缩，相应的元数据将包含在VF中。在某种情况下，Inner Package数据中有多个不同用途的数据，可以对Inner Package数据的每个子区域应用不同的加密或压缩方法，相应的元数据也会是多个

为了减小数据的大小，在TUP生成环境中对能够压缩的数据进行压缩。压缩后的数据和生成的压缩块表(如有必要)将被存储。通过存储压缩块表并使用压缩前的偏移量，可以随机访问压缩后的数据

如果数据加密强度足够，则可以跳过TUP生成环境中的加密。但是，如果无法解密OTA Global Master，可用的操作将受到限制。如果数据需要保密，但没有加密或强度不够，可以在TUP生成环境中进行加密，然后存储。

此外，Inner Package数据及其生成的元数据的ICV将被单独计算和存储。ICV通常期望使用“压缩后加密”的数据进行计算，当需要异常处理顺序时，可以通过生成额外的元数据来扩展行为。

这种结构有助于实现以下目标

[目标设备的目标优化

- 内包的实际数据可以在任何位置启动。因此，当“mmap一个TUP文件并将其内存传递给硬件安全功能”时，可以优化与内存页对齐的数据。
- 如果在使用更新框架功能生成TUP文件时压缩了数据，则可以在目标机上解压缩后再下发到某些ecu。因此，通过透明地解压缩，既可以降低压缩数据传递到目标的成本，又可以减轻资源较少的ecu的负担。

  此外，压缩方法是可选择的，因此可以通过使用目标硬件支持的压缩方法来提高处理效率

-

【目的】数据泄漏预防

需要保密的数据需要加密分发，但目标上可用的安全特性因ecu而异，并且并不总是可用足够强度的加密。对于更新一个自身无法解密数据的ECU，可以对数据进行加密并传递给OTA主服务器，解密后的数据可以通过安全的路由进行传递并进行强保护

[Objectivel Random Accessibility .

如果Inner Packages是压缩和简单存储的，那么为了提取特定的位置数据，必须对整个数据进行解压缩。从传输量和处理成本的角度来看，只需要传送所需数据对应的压缩数据。TUP可以有压缩前后的位置对应表(压缩块表)。通过使用这个表，可以对中间位置的数据进行提取和解密。

对于不需要随机访问的数据(总是需要整体的数据)，可以省略压缩的块表。

目标伤害和抗篡改

如果每个内部包数据使用非防篡改方法(例如XTS)加密，则应生成CV以进行验证，因为数据可能被随机破坏。为了在每个Inner Package数据到达目标ECU时(在OTA master从TUP文件中提取数据之后)进行验证，不应该为整个TUP文件准备验证数据，而应该为每个Inner Package准备验证数据

Inner包元数据

Inner Package的元数据(ICV，压缩块表)的实际数据，如ICV值数组和表，可以分配到TUP文件的任何位置，到达这些位置的信息存储在VF中。

这种结构有助于实现下列目标

【目标】目标设备的优化

根据预期用例，可以选择数据排列。例如

- 每个元数据与其原始数据相邻放置，元数据根据其类型放置在
- 一起，对于特定元数据保证一定的对齐
-

[目标]伤害和抗篡改

每个元数据的实际数据可以用ICV树或ICV阵列进行验证。这些ICV树或ICV数组也可以用上层ICV数组进行验证

目标提高生成文件的处理效率

为了减少高成本元数据的生成时间，可以在不确定实际数据在TUP文件中的位置的情况下获得ICV或压缩块表等元数据。

V可变页脚(VF)

存储元数据的区域(例如用于解压或验证每个区域的数据以及TUP文件本身的元数据)。VF由TLV格式记录组成。

type和offset对的列表可以包含在VF的第一部分，作为TLV，其类型为index'。即使有许多Inner Package，也不需要按顺序检查所有现有的tlv，因为有Inner Package元数据的“索引”信息

这种结构有助于实现下列目标

【目的】防止数据泄露

TUP文件中数据的解密和解压缩信息基本上作为TLV记录包含在VF中。FF中包含VF本身的解密信息，如果对其进行加密。

【目标】Damage and Tamper Resistance

VF区域包含在ICV树的范围内，可以是专用的，也可以是与他人共享的，并且可以防止篡改

FixedFooter (FF)

固定大小的页脚区域，位于TUP文件的末尾。只有在所有其他部分被结构化后才能生成的信息，例如

- 顶部ICV阵列信息VF加密方法
- 导出方法签名签名
-
-

以及随机损伤检测的校验和都存储在这里

如果FF必须加密，则加密信息不能存储在TUP文件中。如果要对数据保密，就必须像FH的情况一样，由环境提供附加信息。

这种结构有助于实现以下目标

目标伤害和抗篡改

- 可以通过从顶部ICV数组递归地验证每个ICV树或ICV数组来检测TUP文件的篡改。通过存储顶部ICV数组本身的ICV，并在FF中使用签名派生方法来保证顶部ICV数组本身的有效性。由于签名是为FF之前的所有数据生成的，因此如果签名被成功验证，则它们是可信的。为了防止TUP即使在目标失去完全控制的情况下也被遗忘，签名必
- 须是与ICV值不同的数据，它只能被验证，而不能在目标上生成(目标上应该没有办法生成它)。由于这个原因，如果一个特定的签名派生方法被破坏，则需要一个与TUP处理系统分离的适当的撤销机制

- 使用ICV进行损伤检测是严格的，但成本很高，因为它需要分析TUP文件。对于可以以足够的准确性确认损坏并且不会对安全产生影响的用例，应该在文件末尾添加一个简单的校验和及其生成方法信息。由于必须能够在没有任何附加信息的情况下验证校验和，因此它不受加密的约束。例如，在调查发生TUP文件损坏的进程时，可以使用此功能。即使是不具备解释up能力的节点，也仍然可以检测到损坏

TLV (Type-Length-Value)格式

需要TUP可扩展性的区域中包含的数据应该具有一般TLV(类型-长度-值)记录的结构。TLV可以嵌套。这意味着某些类型在Value部分有它的子TLV组。是否将V部分解释为TLV由父类型决定。

如果Value部分数据需要有特定的对齐条件，则可以在Length和Value之间填充区域

这一措施有助于实现以下目标

【目标】运营保障

在处理包含可选功能扩展的更新时，解释系统可以安全地忽略未知类型。

由于在TLV类型值范围内保留了专有扩展的区域，因此可以定义在为特定用例扩展时不会干扰标准TUP的附加类型

[目标]目标设备的优化

价值部分可以与目标硬件对齐。考虑到空间成本，在每个TLV记录中都有填充区域是不可取的。然而，为特定TLV类型定义填充的规范也太复杂而难以管理。因此，在Type中保留一个特定的位，如果该位为'1'，则被解释为在Value之前有填充长度和填充的区域。

[目的]防止数据泄漏

通过定义对Value部分进行加密的TLV记录类型，可以为重要数据或元数据提供额外的保护(加密)

CV树/ ICV数组

为了检测篡改和损坏，需要以下几点。

- 可以从数据中生成ICV进行保护，并检查是否与期望值匹配。期望值本身可以用上ICV进
- 行检查。

TUP有两种数据结构，ICV树和ICV数组来存储ICV，以便有效地处理:

- 期望随机存取的连续和大面积区域。非连续且面积不大的区域(如多个ICV树)
-

I简历树

分层ICV集合(ICV树)可以通过以下步骤获得

- 将原始数据分割成一定大小的块，计算每个块的ICV。如果给定的ICV集合大小大于块大小，重复这
- 个过程来计算它的ICV

由于ICV收集的大小可能会很大，这取决于原始数据，

- ICV集合存储在页眉/页脚以外的任何区域
- ICV集合的元数据存储在VF或作为TLV的页眉/页脚以外的区域(类型为ICV树)。

因此，每个Inner Package的ICV树的元数据可能存储在VF中，由于ICV集合本身不存储在VF中，因此可以抑制VF的大小(及其验证成本)。

简历数组

CV数组位于页眉/页脚以外的任何区域，作为类型为“ICV数组”的TLV记录。TLV值是一个包含三个元素的数组，每个元素都是

- TUP文件起始位置TUP文件ICV中的大小
-
-

在TUP文件的某些区域。

该区域预计为:

- TLV记录(ICV树或ICV数组)。不
- 透明的数据。

对于ICV树或ICV数组，在验证TLV后，可以递归地验证受TLV保护的数据。在不透明数据的情况下，可以通过检查其ICV是否与期望值匹配来进行篡改检查。

这种ICV树和ICV数组的数据结构有助于实现以下目标

【目标】随机可及性

在不考虑随机存取的情况下，只要有“整个部分的ICV”就可以实现篡改检测。但是，在这种情况下，需要获得整个文件来检测篡改，并且在只需要访问特定范围的数据的情况下，开销不必要地大。

通过计算一定块大小的每个ICV，并将它们聚合起来创建树形结构，只需要检查包含所需数据的块和覆盖它们的ICV树节点，以验证数据一致性

[Objectivel Optimization for the Target Device

- ICV计算的参数(例如块大小)可以根据目标来选择，它们与ICV一起存储在元数据中。因此，如果目标支持特定的ICV处理方法，则参数可以与其对齐。否则，如果目标无法处理大尺寸的块，则可以调整参数。

  在内存充足的环境中，缓存经过验证的中间ICV可以在使用ICV树检查数据时提高实际访问的性能

- ICV树或ICV数组可以位于与目标的分配约束对齐的任何位置，只要这些能够被引用这些的ICV数组检测到

【目标】运行保障

为了实现ecu之间只传递真正必要的数据的操作，如果目标数据的大小不是块大小的整数倍，则最终块中结束后的区域应为0填充(掩码)。这是因为，如果ICV包含不相关的后续数据，那么在简单提取块大小的数据时，无法独立检查每个数据。

【目标】Damage and Tamper Resistance

TUP文件中需要保护的区域*应该是:

- 由ICV数组直接引用。包含在ICV树的目标范围内。
-

*除不能在FF中使用ICV的数据外的所有数据，以防填充部分也受到保护。

TLV记录以检查这些目标区域为整体构成树形结构。因此，ICV数组或ICV树要么是

- 顶端的ICV数组本身。由上ICV数
- 组指代

通过检查每个TLV本身与上TLV，每个ICV或计算它的参数被保护免受篡改

在ICV数组或ICV树的框架内无法验证顶部ICV数组。因此，FP应该包含验证信息和安全签名，以保证顶部ICV的有效性

[目标]生成文件的处理效率

ICV的加密和生成可以对ICV树或ICV数组的每个目标独立完成，而不是对en一次性全部完成。

安全

为了保证安全性，需要硬件支持。如果存在以下项目，则应有效使用，即使没有硬件支持，也应提供最低保障。

可以存储密钥的内存和存储器。

- 时钟的时间不能被随意篡改加密/解密和签名计算，不会泄露结果和密钥到外面
- 不可逆转的密钥撤销
-

通常期望运行Encrypt-then-MAC’，以便同时实现数据保密性(加密)和身份验证(防止篡改)。因此，应该为加密后的数据定义ICV，但有些硬件可能只支持加密前的ICV计算。TUP也支持非标准处理顺序作为格式，但在运行过程中需要考虑性能和安全强度之间的优先级

这一措施有助于实现以下目标

[目标设备的目标优化

由于可用的安全模块因目标车辆而异，因此TUP文件中使用的加密/解密参数不是固定的，而是应该选择适当的参数。而且，OTA主可以处理、交付和模拟(以尽可能安全的方式)，而不是不能安全处理加密的ecu