采用数字签名技术的可信启动方法研究

杨 霞[1,2]，雷 林[1]，吴新勇[3]，吴开均[2]，桑 楠[1]

(1. 电子科技大学信息与软件工程学院 成都 610054；2. 保密通信国家重点实验室 成都 610041；
3. 四川九洲集团研究院 成都 610041)

【摘要】为保障嵌入式设备系统启动时的可信性，分析了现有可信启动技术对硬件模块严重依赖的现状，结合可信度量和可信链理论，提出了一套基于嵌入式Linux的可信启动方法。该方法以固件IROM作为信任根，利用数字签名和完整性验证技术检查启动实体的完整性和真实性，建立了一条从设备开机到内核启动的信任链。实验结果表明：该方法能有效地验证启动实体的完整性和真实性；与采用硬件模块保护启动实体预期度量值的方法比较，该方法无需增加任何硬件开销便可有效地保护预期度量值；同时保证实体更新时的可信检测。

关 键 词 信任链；数字签名技术；嵌入式Linux；信任根；可信启动技术；可信计算
中图分类号 TP309 **文献标志码** A doi:10.3969/j.issn.1001-0548.2016.02.023

Research on the Trusted-Boot Technology Using Digital Signature Technique

YANG Xia[1,2], LEI Lin[1], WU Xin-yong[3], WU Kai-jun[2], and SANG Nan[1]

(1. School of Information and Software Engineering, University of Electronic Science and Technology of China Chengdu 610054;
2. Science and Technology on Communication Security Laboratory Chengdu 610041;
3. Jezetek Research Chengdu 610041)

Abstract Device booting is a critical step and the foundation of trust for embedded systems. Through analyzing related work we find that most current trusted boot technologies rely heavily on the hardware modules such as trusted platform module (TPM). A new trusted boot method is proposed in this paper for embedded Linux system, which is based on the trusted measurement policy and trust chain mechanism. Firstly, this approach takes the firmware IROM as root of trust, which is used to check the integrity and authenticity of the next booting step like BootLoader. Then the BootLoader do the same to the Kernel. So the chain of trust is established from the top of booting to the Kernel. Using the technology of digital signature and Hash algorithm, we implemented the integrity and authenticity checking for each booting entity. The results show that this method can verify the integrity and authenticity of booting entity, and protect the expected metric easily and effectively without other hardware modules. Besides, it ensures the integrity and authenticity of booting entity when they are updated.

Key words chain of trust; digital signature technique; embedded Linux; root of trust; trusted boot technology; trusted computing

信息技术的高速发展，在带来信息产业空前繁荣的同时，危害信息安全的事件也不断发生，信息安全形势日益严峻。目前有多种保障系统安全的技术，比如系统访问控制、入侵检测技术和防病毒体系等[1]，这些技术大都只能提高系统应用层的安全。一个系统能够正常运行的前提是系统本身是安全的，如果系统在启动之前就由于被篡改或破坏等原因进入一种不可信的状态，则基于该系统而建立的任何安全机制都无法确保系统的可信性。传统嵌入式系统的设计很少考虑系统安全问题，存在较多的安全隐患，如用户可通过刷机进入一种不确定的系统状态，或者内核被恶意程序植入而遭篡改，这些操作都可能给用户带来严重的损失和破坏。

目前，已经有一些研究成果可以做到防止内核等关键信息被篡改。本文将现有的可信启动相关研究方法分为两类：一类是需要增加额外硬件辅助，如在PC机领域添置可信平台模块[2-4](TPM)以及在移动设备终端添置移动可信模块[5-10](mobile trusted

收稿日期：2015－02－03；修回日期：2015－07－08
基金项目：国家核高基项目(M1401060112ZX0103301)；保密通信重点实验室基金(2014006275)
作者简介：杨霞(1978－)，女，博士，副教授，主要从事可信计算、嵌入式系统、操作系统安全增强技术等方面的研究.

module, MTM)这类安全芯片，在安全芯片中存储启动实体初始的预期度量值，启动时将实体加载到内存中，通过比较初始预期度量值和当前计算值的一致性，确定启动过程是否可以安全继续。该方法为了保护可信启动预期度量值，需要增加额外的可信硬件模块，这将增加硬件开销和成本。而对成本控制非常严格的嵌入式设备，大多没有这样的可信模块。因此，该类方法对于嵌入式系统而言，具有很大的局限性。

另一类可信启动方法是设计只读的块设备存储启动实体[11]，启动时从该设备中读取未受更改的实体。因为只读设备中实体在出厂时一次性烧写，所以启动中实体无法被篡改，可保证启动实体的完整性。此方法需要加载的实体一次性烧写，无法实现正常的系统更新。

针对上述问题，本文提出了一种无需增加额外的硬件模块，便可有效保障嵌入式系统启动时可信性的方法。该方法利用完整性验证技术和数字签名技术建立嵌入式可信启动信任链，并可有效地保护预期度量值。这种通过验证签名的技术可以验证启动实体内容的完整性和来源的真实性，简化了硬件设计难度，缩减了开销，在保障可信启动的前提下降低了开发难度，同时做到实体可更新。

1　可信启动模型

在计算机系统中，启动过程是系统一切行为的基础。为创建可信执行环境，首先需要明确可信的定义。可信计算组织(trusted computing group, TCG)将可信定义为[12]：如果一个实体的行为总是以预期的方式朝着预期的目标前进，那么这个实体就是可信的。以TCG定义为基础，本文结合嵌入式系统的特征提出了可信启动模型。

定义 1　每个启动阶段为一个实体E。

定义 2　第一个启动的实体E_0，称为信任根，是可信启动信任的基础。

定义 3　在启动过程中，实体E_{i-1}启动实体E_i，其中$1\leqslant i\leqslant n$。如果启动链中的每一个实体都被验证为可信，则整个启动过程是可信的。在可信启动中，为了确保实体E_i的可信性，必须首先信任E_{i-1}。可信启动的验证模型为：

$$E_{i-1}\xrightarrow{v_{i-1}}E_i\quad(1\leqslant i\leqslant n)\tag{1}$$

式中，v_{i-1}作为一个可信验证模块，用于实体E_{i-1}对E_i的可信性验证。

以信任根为基础逐级验证，最终得到一个可信启动信任链：$E_0\rightarrow E_1\rightarrow E_2\rightarrow\cdots\rightarrow E_n$。

2　采用数字签名技术的可信启动框架

2.1　安全算法介绍

数字签名[13]技术是通过密码技术对电子文档进行电子形式的签名，是维护网络数据安全的重要技术，可用来保证信息内容的完整性和信息来源的真实性。本文采用数字签名技术建立了可信启动信任链，使用哈希算法验证启动实体的完整性，及非对称加密算法RSA保护预期度量值。

2.2　可信启动框架设计

针对嵌入式系统的启动过程，设计如图1所示的可信启动框架。

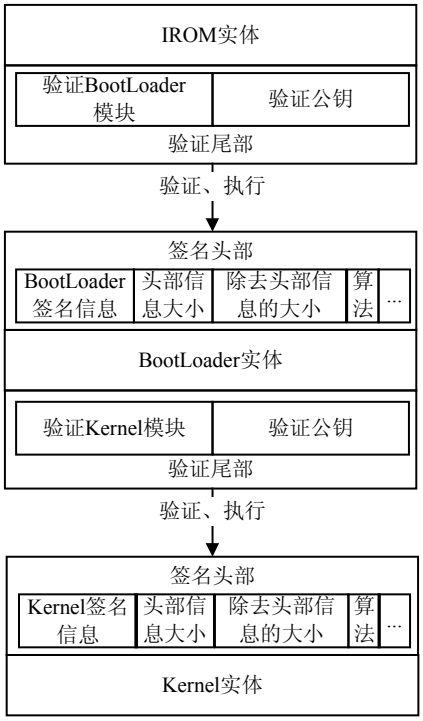

图1　可信启动框架

该框架在正常的嵌入式Linux启动过程中增加了固件IROM，作为信任根。系统上电启动后，IROM为第一个启动实体，然后逐步建立一个可信启动信任链：IROM→BootLoader→Kernel。

在可信启动链的上一个阶段实体的尾部增加一个验证公钥和验证模块；下一个阶段实体的镜像头部增加一个签名头部，头部包含实体自身的签名信息、头部信息大小、除去头部信息后的大小、签名算法等相关参数。上一个启动实体(如IROM)在验证下一个实体(如BootLoader)的可信性和完整性时，由其尾部增加的可信验证模块用“验证公钥”解密下一个启动实体的“签名信息”，得到一个完整性度量

值。接着，将此完整性度量值与当前计算的预启动实体的完整性度量值作对比，如果一致，表明实体完整，准予启动；否则，结束启动过程，并报错。此技术无需额外硬件支持便可以保护可信启动过程中完整性预期度量值的安全性。

2.3 启动实体的签名和验证过程

本文采用数字签名技术对启动实体签名的过程：首先，用非对称加密算法RSA生成一对密钥；其次，用哈希算法对编译好的启动实体镜像制作镜像的摘要，用私钥对摘要值加密；然后，将加密后的密文添加到启动实体镜像的头部作为头部的一部分，生成新的镜像。

镜像启动时的验证过程：上一阶段镜像用公钥解密下一阶段镜像签名头部中的签名信息；上一阶段启动实体计算下一阶段镜像除去头部后内容的摘要；比较摘要和解密后的明文，如果相同，则可证明下一阶段镜像的完整性和来源真实性。为了建立完整的可信启动信任链，需要对BootLoader镜像和Kernel镜像添加签名头部信息，BootLoader镜像和IROM中添加验证尾部，如图1所示。

3 启动实体签名头部的生成

为了保障启动过程实体的完整性、真实性及形成信任链，使用数字签名技术重新制作BootLoader镜像和Kernel镜像，重新生成的镜像能够被上一阶段镜像验证，证明镜像自身的完整性和来源的真实性。

3.1 BootLoader签名头部的生成

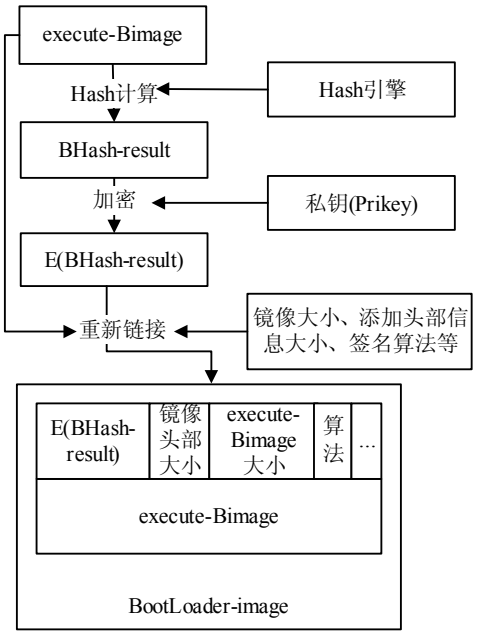

图2 BootLoader镜像添加头部信息的过程

对BootLoader源代码进行编译，生成可执行镜像execute-Bimage。在本地计算机上用公开Hash算法对镜像execute-Bimage进行Hash计算，得到镜像的哈希值BHash-result，然后用受信方提供的与IROM验证代码中存放的公钥相匹配的私钥对BHash-result进行加密，得到加密后的值E(BHash-result)，将E(BHash-result)、镜像大小、头部大小、Hash算法、RSA算法等信息与execute-Bimage重新链接生成镜像BootLoader-image，BootLoader-image为最终生成的BootLoader镜像。添加头部信息流程如图2所示。

3.2 Kernel签名头部的生成

与上面为BootLoader镜像添加头部信息流程一样，为Kernel镜像添加头部信息过程中生成Kernel原始镜像execute-Kimage、哈希值KHash-result、加密后的值E(KHash-result)和最终镜像Kernel-image。

4 构建可信启动信任根

IROM作为可信根，在出厂时一次性烧写之后无法被更改，以此保障其自身的可信性。当芯片上电启动时，最先跳转到IROM中执行，由它对BootLoader进行完整性检测。从信任根开始，逐级向下检测，通过检测后执行下一阶段镜像，从而完成整个可信启动流程。

IROM作为可信启动最初的启动镜像，具有CPU最初的执行权，主要功能是简单硬件初始化及验证相关镜像：初始化系统时钟和设备控制器，以及准备启动设备并验证下一阶段镜像的真实性和完整性。IROM验证BootLoader镜像，如果验证通过，则跳转到BootLoader在内存的起始地址开始处执行，将CPU的执行权交给BootLoader；如果验证不通过，则执行相关验证失败操作，如关机等。该镜像不涉及被检验，制作时可以直接通过源码编译生成，然后将生成的镜像文件在出厂前烧写即可。

5 可信启动验证模块的实现

根据信任链传递思想：先度量，再验证，最后跳转[14]。如图1在IROM、BootLoader的尾部添加了验证模块和验证公钥两个部分。验证模块采用数字签名验证技术对启动实体的可信性进行验证，确保启动实体的来源可信。为了满足系统各阶段不同安全级别的要求，分别在IROM、BootLoader中的验证模块和验证公钥中使用不同的Hash算法和不同密钥长度的RSA算法。上一阶段实体的验证公钥需和下一阶段实体头部中的签名信息生成时的私钥配对。

可信启动分为IROM验证BootLoader和由BootLoader验证内核两个阶段，各个阶段的验证过程类似，此处仅以第二阶段为例进行阐述。

BootLoader验证内核的过程分为如下几个步骤：

1) 首先将内核镜像拷贝到内存中，然后根据镜像Kernel-image头部信息解析出E(KHash-result)与execute-Kimage。

2) 使用与制作内核镜像时用到的私钥相匹配的公钥解密E(KHash-result)，根据非对称加密算法原理，如果能够解密，则说明该镜像头部中E(KHash-result)为可信的，同时得到解密的结果KHash-result。如果不能解密，则说明该镜像来源不合法，镜像不可信，将执行验证失败操作。

3) 同时对镜像Kernel-image的尾部execute-Kimage用与制作Kernel镜像时用到的Hash算法相同的算法进行计算，得到重新计算的哈希值KHash-result-T。

4) 如果KHash-result-T与KHash-result相等，则说明execute-Kimage的内容是完整的，因此可以确定整个镜像的来源真实，内容完整；如果不相等，则说明该镜像遭到篡改，不具备完整性特性，将执行验证失败操作。验证过程如图3所示。

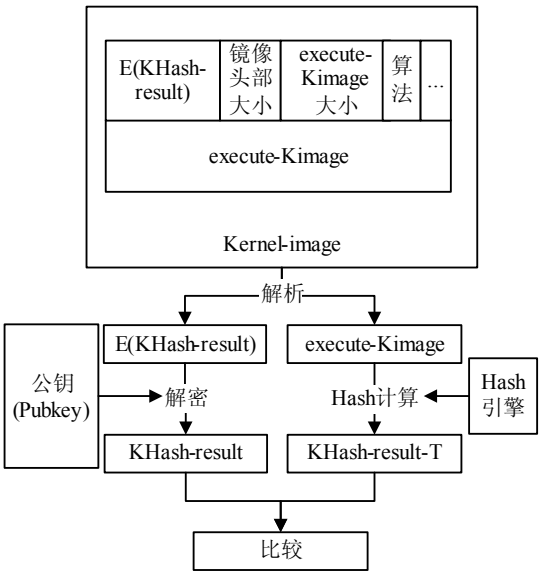

图3　IROM对BootLoader的可信验证

经过以上两阶段的验证，保证了从固件IROM到Kernel的整个过程的可信，即建立了整个启动过程完整的信任链。

6　实验原型系统

实验系统的硬件平台为天嵌2440嵌入式开发板；软件平台BootLoader为U-Boot1.1.6，内核为Linux2.6.30。采用SHA-1哈希算法和2 048位密钥的RSA算法实现系统启动时的可信验证。本文对两个可信启动阶段都进行了反复50次以上的测试，两个阶段均符合预期，此处同样以第二阶段为例进行实验和测试。

6.1　合法内核的可信启动测试

图4为U-boot验证一个合法的Kernel-image镜像，图中打印了镜像的头部签名信息E(KHash-result)、除头部信息后内容的SHA-1值KHash-result-T和解密签名信息后的结果KHash-result。图中解密得到了KHash-result值，说明镜像头部的签名信息真实完整，再有KHash-result和KHash-result-T两值相等，说明exeute-Kimage内容和头部签名信息匹配。所以该Kernel镜像是合法的镜像，结果成功启动。

```
.Copy Kernel to SDRAM done
.E(KHash-result) (from the head of Kernel-image):
196B0585002037A7EFB47AC9A490846A245C1AF413F1D5D8E341B5
4309A375353BE158CC397C2516E35B5C9C207A27DF6B4417F44D97
4518E1832AD46C5C79B40A2741DB437AD3A8218AA4F8F01641D556
6602B6CACF2D2DCB213C0EC5C7B9C477B808C42982BD44A465677F
EFE451AFC71B78C719E97E3ECB5FEAEB31A1ADD702F1C9A5C899E0
AC1A10F4F8C83480DAC41F805D59B95B7D4E62D057C17B4E83FE07
BC1D0E4A566EF0901A75919695BABC19B07BBC23D47660813F4F10
81AEF9DA64141B3CAFB5E9AECC2172688A24F5A38AAB55BF7AD5D7
ACCD752D7FBE965BA1B96907D2ACF2AA03C686A0E92E30B3E7AE25
FD78D0F43E0B41B2B8660A9093
.Verifying
.KHash-result-T:FF46C0886146D6BE08F4FBF580CB5332DB303284
.KHash-result:FF46C0886146D6BE08F4FBF580CB5332DB303284
.OK
NOW, Booting Linux......
```

图4　合法Kernel-image的验证测试

6.2　非法内核的可信启动测试

图5为非法的Kernel-image镜像启动信息。

```
.Copy Kernel to SDRAM done
.E(KHash-result) (from the head of Kernel-image):
34780585002037A7EFB47AC9A490846A245C1AF413F1D5D8E341B5
4309A375353BE158CC397C2516E35B5C9C207A27DF6B4417F44D97
4518E1832AD46C5C79B40A2741DB437AD3A8218AA4F8F01641D556
6602B6CACF2D2DCB213C0EC5C7B9C477B808C42982BD44A465677F
EFE451AFC71B78C719E97E3ECB5FEAEB31A1ADD702F1C9A5C899E0
AC1A10F4F8C83480DAC41F805D59B95B7D4E62D057C17B4E83FE07
BC1D0E4A566EF0901A75919695BABC19B07BBC23D47660813F4F10
81AEF9DA64141B3CAFB5E9AECC2172688A24F5A38AAB55BF7AD5D7
ACCD752D7FBE965BA1B96907D2ACF2AA03C686A0E92E30B3E7AE25
FD78D0F43E0B41B2B8660A9093
.Verifying
.KHash-result-T:FF46C0886146D6BE08F4FBF580CB5332DB303284
.Kernel-image was Destoryed
.Fail
```

a. E(KHash-result)被篡改

```
.Verifying
.KHash-result-T:497C665E549B0CB55517B04F1565386D92FA6E42
.KHash-result:FF46C0886146D6BE08F4FBF580CB5332DB303284
.Kernel-image was Destoryed
.Fail
```

b. exeute-kimage被篡改

```
.Verifying
.KHash-result-T:763DB18B58BA26E23A91C598673FEDEA2ACD01E4
.Kernel-image was Destoryed
.Fail
```

c, E(KHash-result)和exeute-kimage被篡改

图5　Kernel-image被篡改后的验证测试

同图4比较，图5a、5c中无法解密得到Kernel-result值，说明头部签名信息被篡改；图5b中KHash-

result值未发生改变，说明头部签名信息完整。KHash-result-T值发生改变，说明exeute-Kimage内容发生改变。KHash-result值和KHash-result-T值不相等，说明exeute-Kimage内容和头部的签名信息不匹配。此处的3种情况都为非法镜像，故启动失败。

6.3 内核更新时的可信启动测试

同图4比较，图6中E(KHash-result)值发生改变，同时KHash-result值能够被解密得到，说明此时的E(KHash-result)值为新的镜像的合法签名信息；KHash-result-T值和KHash-result值相同，说明exeute-Kimage内容和头部签名信息匹配。所以此镜像是合法的镜像，成功启动，升级成功。

图6 Kernel-image升级时可信验证

7 结 束 语

本文针对嵌入式系统所面临的安全威胁，提出了一种采用数字签名技术的可信启动方法，以嵌入式Linux操作系统为平台构建了一个可信启动信任链。该方法无需增加额外的硬件设备(如TPM、MTM等)，通过数字签名技术保护实体的预期完整性度量值，确保启动实体的完整性和来源的真实性，防止关键系统软件被篡改，为应用程序搭建可信的执行环境。此外，该方法还可实现系统更新，并确保更新内容来源的可信性。由于该方法无需增加任何的硬件成本，便可有效地保证系统启动过程的可信性，因此可广泛应用于严格控制成本的嵌入式系统开发中，以提高系统的高可信能力。

参 考 文 献

[1] 冯登国. 可信计算理论与实践[M]. 北京: 清华大学出版社, 2013.
FENG Deng-guo. Trusted computing theory and practice[M]. Beijing: Tsinghua University Press, 2013.

[2] ASOKAN N, EKBERG J E, KOSTIAINEN K, et al. Mobile trusted computing[J]. Proceedings of the IEEE, 2014, 102(8): 1189-1206.

[3] YU Chao, YUAN Men-ting. Security Bootstrap based on trusted computing[C]//2010 Second International Conference on Networks Security Wireless Communications and Trusted Computing. Washington D C, USA: IEEE Computer Society, 2010: 486-489.

[4] YU Fa-jiang, ZHANG Huan-guo. Design and implementation of a bootstrap trust chain[J]. Wuhan University Journal of Natural Sciences, 2006, 11(6): 1449-1452.

[5] Trusted Computing Group. TCG mobile trusted module specification, specification version 1.0, revision 6[EB/OL]. [2014-11-26]. http://www.trustedcomputinggroup.org/files/resource_files/87852F33-1D09-3519-AD0C0F141CC6B10D/Revision_6-tcg-mobile-trusted-module-1_0pdf.

[6] KAI Tang, XIN Xu, GUO Chun-xia . The secure boot of embedded system based on mobile trusted module[C]//2012 International Conference on Intelligent System Design and Engineering Application. New York, USA: IEEE, 2012: 1131-1134.

[7] 赵波, 费永康, 向骕, 等. 嵌入式系统的安全启动机制研究与实现[J]. 计算机工程与应用, 2014, 50(10): 72-77.
ZHAO Bo, FEI Yong-kang, XIANG Shuang, et al. Research and implementation of secure boot mechanism for embedded systems[J]. Computer Engineering and Applications, 2014, 50(10): 72-77.

[8] 张焕国, 李晶, 潘丹铃, 等. 嵌入式系统可信平台模块研究[J]. 计算机研究与发展, 2011, 48(7): 1269-1278.
ZHANG Huan-guo, LI Jing, PAN Dan-ling, et al. Trusted platform module in embedded system[J]. Journal of Computer Research and Development, 2011, 48(7): 1269-1278.

[9] PARNO B, MCCUNE J M, PERRIG A. Bootstrapping trust in commodity computers[C]//2010 IEEE Symposium on Security and Privacy. Oakland, USA: IEEE, 2010: 414-429.

[10] KHALID O, ROLFES C, IBING A. On implementing trusted boot for embedded systems[C]//2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). New York, USA: IEEE, 2013: 75-80.

[11] CLARK P C, HOFFMAN L J. A smartcard protected operating system[J]. Communications of the ACM, 1994, 37(11): 67-70.

[12] Trusted Computing Platform Alliance. TCG PC specific implementation specification, Version1.1[EB/OL]. [2014-12-10]. http://www.trustedcomputinggroup.org/files/resource_files/87B92DAF-1D093519AD80984BBE62D62D/TCG_PCSpecificSpecification_v1_1.pdf .

[13] SHEN Chang-xiang, ZHANG Huang-guo, FENG Deng-guo, et al. Survey of information security[J]. Science in China Series F: Information Seiences, 2007, 50(3): 273-298.

[14] FENG Deng-guo, QIN Yu, FENG Wei, et al. The theory and practice in the evolution of trusted computing[J]. Chinese Science Bulletin, 2014, 59(32): 4173-4189.

编 辑 蒋 晓