计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210300237

面向 IOT 芯片的安全启动算法分析与应用

宗思洁　覃　天　贺龙兵
东南大学微电子学院　南京 210096
(220184855@seu.edu.cn)

摘　要　RSA 和 ECC 算法是目前应用于 IOT(物联网)邻域的公钥加密算法标准。文中通过对比 RSA 和 ECC 算法的性能及安全性,优选 ECC 算法作为 IOT 应用的安全启动算法,提出了应用于 IOT 芯片启动的安全驱动解决方案。通过在 IOT 芯片上进行启动验证,结果表明,ECC 算法可以应用于 IOT 芯片的启动,同时成本、性能、安全性表现更优。文中所研究的内容为物联网安全提供了解决方案。

关键词:系统启动;椭圆加密算法;物联网安全;RSA 算法
中图法分类号　TP309

Analysis and Application of Secure Boot Algorithm Based on IOT Chip

ZONG Si-jie,QIN Tian and HE Long-bing
School of Microelectronics,Southeast University,Nanjing 210096,China

Abstract　RSA and ECC are currently standard public key encryption algorithm in IOT chips. By comparing the performance and security between RSA and ECC algorithms,ECC is found to be more suitable for IOT applications. A secure driver program is proposed as the solution of the secure startup of IOT chips. Experimental verification based on IOT chips demonstrates that ECC algorithm possesses several advantages including lower cost,higher performance,and better security. This paper provides a solution for the security development of the Internet of Things.

Keywords　Startup,Elliptic curve cryptography,IOT security,RSA

1　引言

IOT 芯片是应用于特定物联网场景的一类集成电路(IC)。现今,物联网应用场景多种多样,对芯片的整体需求量很大,但针对同款或同系列产品 IOT 芯片的需求量相比传统 IC 芯片依然少很多。因此,成本控制是 IOT 芯片生产中的重要方面。此外,随着万物互联时代的到来,端对端的连接密度越来越大,终端数量大幅增长[1-2],终端之间通信交流的激增也带来了越来越多的安全隐患问题[3-4]。安全可信已成为评价 IOT 芯片可靠性的重要指标[5]。

对于物联网应用来说,应整体上从芯片、嵌入式软件以及应用接口 3 个层面综合考虑安全性以及相应防护措施,从而保证整个系统安全,而不能只对独立芯片进行安全评估。当前,国内外已有许多针对系统安全的研究,如可信平台模块(TPM)[6-7]、信任链的设计以及可信运行环境(TEE)等,这些都是从系统角度来保证 IOT 应用安全问题。

本文对 RSA[8] 和 ECC[9] 算法的性能、成本以及安全性进行分析比较,提出了一种应用于 IOT 芯片的安全启动方案。本文首先介绍 RSA 和 ECC 算法的原理,阐述传统的系统启动过程;随后对比分析,选择一种更适用于 IOT 芯片的安全启动算法并分析算法在系统安全启动过程中的作用,从成本、性能、安全 3 方面进行了综合评估。本文所研究的内容为物联网安全发展提供了解决方案。

2　安全启动算法分析

在系统安全启动过程中,安全驱动的作用是对数据进行加解密从而保证数据的安全和可靠性,同时,安全驱动作为可信启动的度量工具,在启动和度量可信过程中还承担着仲裁器的作用。因而,选择成熟且可靠的标准密码技术才能保证芯片的安全使用。安全驱动中包含的摘要算法一般是 sha256,对称加密算法则选择 AES,验签算法使用 RSA 或 ECC 算法。RSA 是相当成熟的公钥加密算法,但其密钥长度长,不便于密钥管理[10];ECC 算法可以用较短的密钥长度达到和 RSA 相同的安全强度,但其速度和性能有待加强。对于 IOT 芯片而言,选择何种验签算法需要综合分析与比较。本节首先介绍 RSA 和 ECC 算法的原理及步骤。

2.1　RSA 加密算法简介

2.1.1　RSA 概述

RSA 是由 Rivest 等于 1977 年提出的一种非对称加密算法。它的特色在于双密钥机制,即有一对密钥(公钥,私钥)。其中公钥是任何人可见的,而私钥则是要获取正确消息的人所保护起来的。其中,公钥用于加密,私钥用于解密。为了确保发送消息的正确性,RSA 还可以进行私钥签名,公钥解签,如图 1 所示。

基金项目:航空科学基金项目(ASFC-20170269003)
This work was supported by the Aeronautical Science Foundation of China(ASFC-20170269003).
通信作者:贺龙兵(heloongbing@seu.edu.cn)

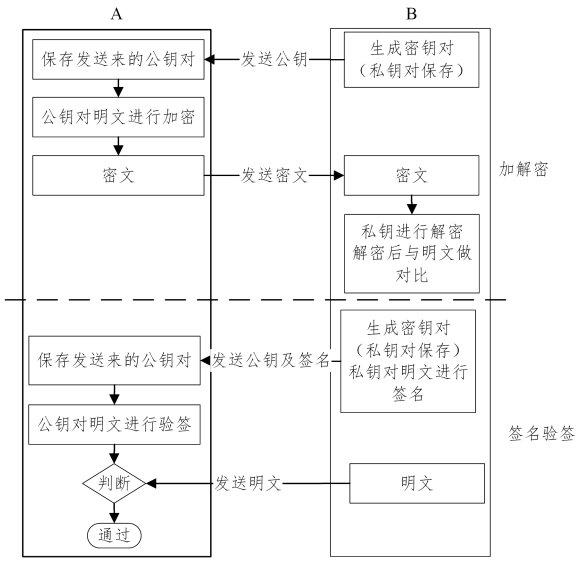

图 1　RSA 加解密流程

Fig. 1　RSA algorithm flow

2.1.2　RSA 加解密流程

RSA 的密钥生成步骤如下：随机选择两个不相等的大质数 p 和 q，一般是由随机数生成器生成，计算 p 和 q 的乘积 n，计算 n 的欧拉函数 $\varphi(n)=(p-1)(q-1)$，随机选择一个整数 e，条件是 $1<e<\varphi(n)$，且 e 与 $\varphi(n)$ 互质。计算 e 对于 $\varphi(n)$ 的模反元素 d：$ed \equiv 1 \pmod{\varphi(n)}$，将 n 和 e 封装成公钥，n 和 d 封装成私钥。加密计算公式为：$c \equiv m^e \pmod{n}$，解密公式为 $m \equiv c^d \pmod{n}$，其中 m 为输入明文，c 为密文。

RSA 的加解密流程如下：1）将输入数据 M 进行填充，填充的方式有两种：RSAES-PKCS1-v1_5 和 RSAES-OAEP。填充是对输入数据的一些处理，RSAES-PKCS1-v1_5 是早期 RSA 的填充方式，较为简单且可靠性较低，目前使用非常少，因此本文介绍 RSAES-OAEP 方式的填充，如图 2 所示。2）填充完毕之后将得出的 EM 值进行公钥加密计算。3）将公钥加密计算的数据进行私钥解密。4）将解密的数据进行解填充。若解填充之后的数据与原来数据一致则说明此过程是可靠的。

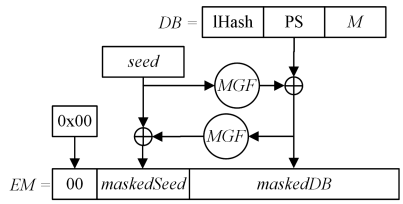

图 2　RSAES-OAEP 填充方法流程图

Fig. 2　RSAES-OAEP padding flowchart

填充步骤：1）产生全零的 PS 字符串，长度为（N-$mlen$-$2hlen$-2），数值有可能为 0，其中 N 为模数 n 的长度，$mlen$ 为消息长度，$hlen$ 为所选摘要算法输出摘要的长度。比如选择 sha256，输出长度为 32 字节。2）计算 lHash＝HASH(L)，其中 L 为与消息摘要相关的标签，lHash 的长度为 $hlen$。3）将 lHash 和 PS 加上一个固定字节的 0x01 再加上消息拼接成 DB，生成长度为 $hlen$ 的随机 $seed$。4）计算 $mgfSeed = MGF(seed, hlen)$，其中 MGF 为掩码生成函数。5）计算 $maskedDB = maskedSeed$ xor DB，xor 是异或运算。6）计算 $mgfDB = MGF(maskedDB, N\text{-}hlen\text{-}1)$。7）计算 $maskedSeed = seed$ xor $mgfDB$。8）将固定字节 0x00，$maskedSeed$，$maskedDB$ 拼接起来构成填充后的消息 EM，长度为 K 字节。

2.1.3　RSA 签名验签流程

RSA 的签名验签流程如下：1）将输入数据 M 进行填充。填充的方式有两种：RSASSA_PKCS1-v1_5 和 RSAES-PSS。2）填充完毕之后再将得出的 EM 值进行私钥签名计算。3）将私钥签名计算的数据进行公钥验签。4）将解密的数据进行解填充。若解填充数据与原始数据一致，则说明此过程是可靠的。图 3 给出了以 RSAES-PSS 方式解填充数据的判断流程，其中 H 是 M' 的哈希值，$emBits$ 值等于数据的比特长度减 1，$emlen$ 是 $emBits$ 值除以 8 再向上取整。通过对比解签数据解填充之后的 H 值和填充时计算得到的 H 值的一致性来保证签名验签流程的正确性。

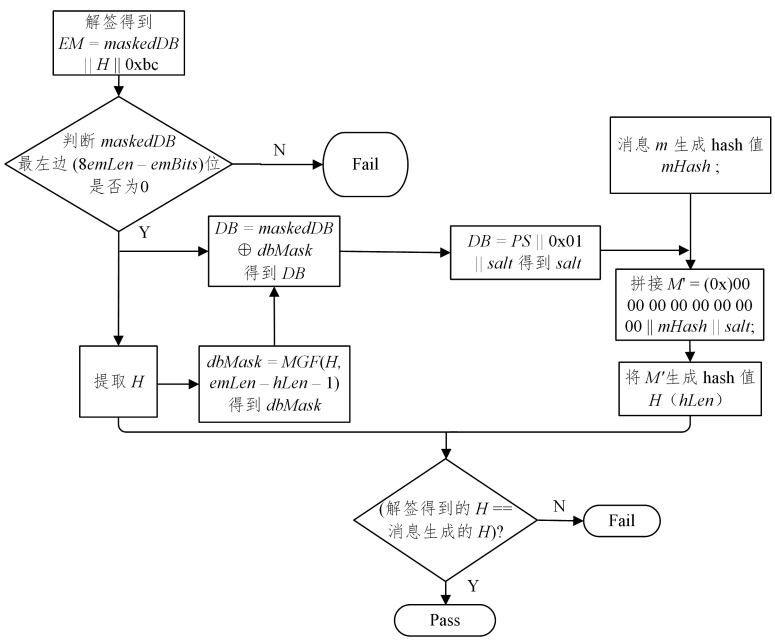

图 3　RSAES-PSS 解填充比较

Fig. 3　RSAES-PSS de-padding and checkflow

2.2 ECC 加密算法简介

2.2.1 ECC 概述

ECC 是一种基于椭圆曲线离散对数的非对称加密算法。与 RSA 相比，其优势是在相同长度密钥下，ECC 算法可以达到更高的安全性。ECC 算法主要应用的机制也是公钥加密、私钥解密，私钥签名、公钥解签。2010 年 12 月中国国家密码管理局颁布的国密算法 SM2 就是基于 ECC 算法。

ECC 算法最初由 Neal Koblitz 和 Vector Miller 两人于 1985 年提出，其数学依据是构建了有限域上的椭圆曲线离散对数问题(ECDLP)。有限域是指域中有有限个元素，域中所包含的元素个数称为域的阶。ECDLP 比因子分解问题更难，其难度是指数级变化。

椭圆曲线一般用二元三次方程表示。大部分的二元三次方程都可以简化为维尔斯特拉斯方程：$y^2=x^3+ax+b$，其中 $a,b\in R$ 且需满足条件 $4a^3+27b^2\neq 0$。该条件保证方程中不会出现奇异点从而获得平滑的椭圆曲线。同时，由于实数域的椭圆曲线一般不适合构建密码机制，因此，为了把椭圆曲线上的点离散化，需要将椭圆曲线定义到有限域中，即 $y^2=(x^3+ax+b) \bmod P$，其中 $p>3$ 且 p 为素数，$a,b\in Z_p$ 且满足 $4a^3+27b^2\neq 0$。密码算法通常运行于具有某些保持封闭性的代数结构空间中，这种代数结构就是有限循环群。有限循环群具有如下特性：1)封闭性，即运算的结果也在这个群中；2)结合律，即 $a+b+c=a+(b+c)=(a+b)+c$；3)存在单位元，即对于任意 a，有 $a+0=a$；4)存在逆元，即对于任意 a，存在 b 使 $a+b=0$；5)交换律，即元素集合满足 $a\cdot b=b\cdot a$。

确定一条 ECC 的曲线标准需要选择特定的“种子”，这些“种子”决定了 ECC 算法的安全性。目前有如下比较流行的标准：1)美国国家标准技术研究所(NIST)发布的 NIST P-192，NIST P-224，NIST P-384，NIST P-521，Curve25519，Curve448 系列算法，对应的标准是 RFC7748 和 FIPS PUB 186-4。2)高效密码技术标准组(SECG)制定的 secp192r1，secp224r1，secp256r1，secp384r1，secp521r1，对应的标准是 RFC4492。3)ECC Brainpool 制定的 brainpoolP256r1，brainpoolP320r1，brainpoolP384r1，brainpoolP512r1，对应的标准是 RFC 5639。总之，不同的组织发布了不同的椭圆曲线标准，虽然它们都是基于 ECC 的离散对数问题，但是由于参数不同，所以一般不能兼容。比如 NIST 标准下的 P 系列和 Curve 系列的曲线也不能兼容。在名称上一般也有区分，比如 ECDSA 表示的是 P 系列的曲线，EDDSA 表示的是 Curve 系列的曲线。

目前 ECC 的应用主要有基于椭圆曲线的密钥交换算法(ECDH)、椭圆曲线数字签名算法(ECDSA)和椭圆曲线集成加密算法(ECIES)。以上算法均需要明确知道参数 p,a,b,G,n,h，其代表含义如表 1 所列。

表 1 椭圆曲线参数

Table 1 Elliptic curve parameters

参数	描述
p	有限域的大素数
a	整数，椭圆曲线方程系数
b	整数，椭圆曲线方程系数
G	生成元，在椭圆曲线上的某个点
n	素数，椭圆的阶，群内个数
h	余因数

2.2.2 基于椭圆曲线集成加密算法(ECIES)

ECC 应用于加解密的算法称为椭圆曲线集成加密算法(ECIES)。其加解密流程为：1)对象 A 选定一条椭圆曲线 $E_p(a,b)$ 并取曲线上一点作为基点 G；2)对象 A 选择一个私钥 k，并生成公钥 $K=kG$，并将 $E_p(a,b)$，k 和 G 发送给对象 B；3)对象 B 收到数据后将明文编码到 $E_p(a,b)$ 上的一点 M 并产生一个随机数 r；4)对象 B 计算点 C_1 和 C_2($C_1=M+rK$，$C_2=rG$)，再将 C_1 和 C_2 传给对象 A；5)对象 A 计算 $C_1-kC_2=M+rkG-krG=M$，并对 M 解码得到明文。对于上述加解密过程，攻击者通常只能得到 $E_p(a,b)$，G，K，C_1，C_2，因为没有 k 从而无法得到 M，因此可保证算法的安全性。

2.2.3 基于椭圆曲线的密钥交换算法(ECDH)

密钥交换协议(DH)是基于离散对数数学原理提出的一种加密算法，离散对数的公式为 $a=g^i \bmod P$，其中 P 是一个素数，i 取值于 1～($P-1$)之间。如果算出来的 a 各不相同，那么 g 就是 P 的原根。因此对于 g 和 P，a 和 i 有一一对应的关系。离散对数的复杂性在于，根据 g，i 和 P，很容易计算出 a，但是根据 g，a 和 P，很难计算出 i，尤其是在 P 取值非常大的情况下。

DH 密钥交换协议的流程为：1)甲和乙共享一个素数 P 和一个原根 g；2)甲选一个随机数 a，满足 $0<a<P$，同时甲计算出 $A=g^a \bmod P$；3)甲将 A 发给乙，乙选一个随机数 b，满足 $0<b<P$，同时乙计算出 $B=g^b \bmod P$；4)乙将 B 发给甲，甲可以根据 a 和乙发过来的 B 计算出 $K'=B^a \bmod P=(g^b)^a \bmod P$，同时乙可以根据 b 和甲发过来的 A，计算出 $K=A^b \bmod P=(g^a)^b \bmod P$。其总体流程如图 4 所示。可以看出，甲计算得到的 K' 和乙计算得到的 K 是一致的，但甲乙交换的只有 A 和 B，根据 A 和 B 是无法直接计算出密钥 K 的，因而保证了密钥安全。

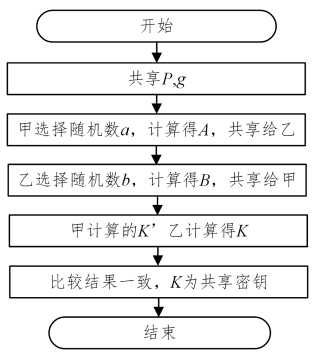

图 4 密钥交换流程图

Fig. 4 DH flowchart

基于 ECC 算法密钥生成步骤：1)对象 A 选定一条椭圆曲线 $E_p(a,b)$，并取曲线上一点作为基点 G；2)对象 A 选择一个私钥 k_1，并生成公钥 $K_1=k_1G$，对象 A 将公钥 K_1 发给对象 B；3)对象 B 选择一个私钥 k_2，并生成公钥 $K_2=k_2G$，然后生成共享密钥 $K=k_2K_1=k_2(k_1G)$；4)对象 B 将公钥 K_2 发给对象 A，对象 A 计算共享密钥 $K=k_1K_2=k_1(k_2G)$。密钥协商成功，后续可以使用 K 进行数据加解密，整个协商过程中其他对象只能获取到 $E_p(a,b)$，G，K_1 和 K_2，无法获取 k_1 和 k_2。因此，其他对象无法获取或生成密钥 K。密钥交换的过程与生成 ECC 公钥私钥对的过程是类似的，都是做了 ECC 点乘运算，只不过生成密钥对时，基点是 G，乘数是私钥 k，生成公钥

$K=kG$；而在密钥交换时，基点是对方的公钥 K_2，乘数是自己的私钥 k_1，生成共享密钥 $K=k_1K_2$。

2.2.4 基于椭圆曲线的数字签名算法(ECDSA)

DSA 又称数字签名算法，是一种用于数字签名和认证的加密算法。DSA 是 Schnorr 和 ElGamal 签名算法的变种，被美国 NIST 作为数字签名标准(DSS)。DSA 加密算法主要依据整数有限域离散对数难题。其要求素数 P 必须足够大，且 $P-1$ 至少包含一个大素数因子以抵抗 Pohlig&Hellman 算法的攻击。DSA 算法流程是发送者使用私钥对数据进行签名，接收者在接到数据之后用发送者分享的公钥来验证签名的真实性。

基于 ECC 算法的签名验签流程是：1)对象 A 选定一条椭圆曲线 $E_p(a,b)$并取曲线上一点作为基点 G；2)对象 A 选择一个私钥 d 并生成公钥 $Q=dG$；3)对象 A 产生一个随机数 k，计算 $R(x,y)=kG$，其中 r 为 R 的横坐标，同时，对象 A 计算输入数据 M 的哈希值 $hash=SHA(M)$；4)对象 A 计算 $s=k^{-1}(hash+rd) \bmod P$；5)对象 A 将 r,s 作为签名数据发给对象 B；6)对象 B 获得 $r,s,E_p(a,b),G,Q$；7)对象 B 计算输入数据 M 的哈希值 $hash=SHA(M)$；8)对象 B 计算 $R=s^{-1}(Ghash+rQ)=k(hash+rd)^{-1}(Ghash+rdG)=kG=R(x,y)$。取 $v=R_X$，若 $v=r$，则验签成功。

3 基于 mbedtls 协议的算法性能分析

3.1 mbedtls 简介

ARM 公司开源和维护的 mbedtls 软件包实质上是一个 SSL/TLS 算法库，它用最小的编码占用空间实现了各种加密算法。mbedtls 的小巧灵活和易于使用的特点使得其可以轻松地在嵌入式产品中加入物联网安全功能。mbedtls 提供的安全加密组件相对而言比较独立，通过单个配置文件配置单个功能就能加入应用中，同时它还具有多种多样的配置选项，代码开发人员可以灵活地按照实际情况使用和裁剪代码。另外，它还包含一个完备的抽象层实现，这个抽象层提高了代码的复用程度，降低了开发难度，mbedtls 还拥有许许多多精心设计的测试用例，从而保证了可靠性和稳定性。

3.2 mbedtls 下 RSA 和 ECC 的性能分析

应用 mbedtls 开源算法对公钥加密算法的性能进行测试，测试平台主要参数为：CPU 主频 2001 MHz，内核数量 1。测试结果如表 2 所列。

表 2 RSA 和 ECC 签名验签速度

Table 2 RSA and ECC signature and verification speed

(单位：times/s)

算法	签名	验签
RSA-2048	349	16 849
RSA-4096	56	4 624
ECDSA-secp521r1	440	118
ECDSA-brainpoolP521r1	80	18
ECDSA-secp384r1	573	153
ECDSA-brainpoolP384r1	169	40
ECDSA-secp256r1	862	243
ECDSA-secp256k1	1 112	300
ECDSA-brainpoolP256r1	318	75
ECDSA-secp224r1	1296	340
ECDSA-secp224k1	1204	311
ECDSA-secp192r1	1917	518
ECDSA-secp192k1	1 698	442

由表 2 可知：1) RSA 的验签速度明显优于签名速度，并且，验签消耗的内存比签名少，因此，RSA 适用于反复验签的场合；2)对于 DSA 算法而言，对于相同椭圆曲线，签名速度比验签速度快，同时 ECC 算法用 256 比特的密钥长度(如曲线 ECDSA-secp256r1 对应算法)就可以达到 RSA 密钥长度为 2048 比特的安全性，并且对于 ECC 算法而言，相同安全性下的签名速度比 RSA 快。另外，ECDSA 算法的最大时间消耗是在点乘操作，进行优化之后速度会有明显提高。

总而言之，对于物联网上的应用来说，尽管 RSA 更成熟，但其密码对组成复杂并会导致密钥管理资源浪费，因此为降低资源占用并保证安全性，应优先选择 ECC 算法作为 IOT 芯片的安全启动算法。

4 安全启动算法的应用与验证

安全驱动一般是包含了各种安全算法的驱动，其中的算法根据具体需求进行选择。在系统启动的过程中验签算法使用频次最多。通过上文比较，ECC 算法更适合于 IOT 应用。在此基础上，本文提出了一种应用于 IOT 芯片的安全驱动优化方案，下文对该驱动方案及启动过程中如何调用 ECC 算法做详细介绍。

4.1 安全驱动的设计与应用

图 5 给出了系统启动的引导流程，其中 FIXED ROM 是将安全驱动固化到 ROM 中的模块，因此，安全驱动只在物联网操作系统中提供可供用户使用的 API 接口。该接口不仅在系统启动中多次调用，而且在用户或者场景需要时也会直接调用。BOOTROM 和 FIXED ROM 是已经固化在芯片内的 ROM，用来引导其他 BOOT，其中 FIXED ROM 是上文已经提到的固化的安全驱动。LOADER 是芯片自启动模块，用于开发或者更新，与 PC 经串口通过 Ymodem 协议进行数据传输。FLASHBOOT 是预先存储在 FLASH 里面的各种 BOOT，KERNEL 是内核模块。

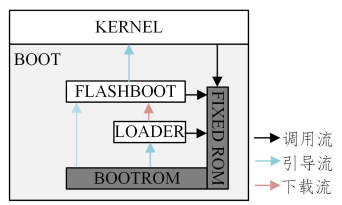

图 5 系统启动模块

Fig. 5 System startup module

芯片上电后，首先运行的第一段代码即为 BOOTROM 里面的代码，其主要的操作是进行硬件初始化，验证存储在 FLASH 里面的公钥，然后进行自启动的判断，若是程序开发阶段，则选择自启动流程，若是正常启动阶段，则引导 BOOT 启动。引导 BOOT 的流程主要是通过 FLASH 中的公钥对 BOOT 头区代码的验证，将验证的签名信息存储到头区中，再由头区的签名数据验证 BOOT 代码区，验证过程所用的算法是 ECDSA。BOOT 启动之后，引导操作系统，操作系统运行之后就进入操作界面。

本方案目前已实现安全驱动固化并且能在系统启动中正常运行，安全驱动的调用如图 4 中黑色箭头所示，在 BOOTROM 引导 FLASHBOOT 启动和 FLASHBOOT 内部代码的运行时，安全驱动会根据加解密、签名验签的需要而被调度使用。

4.2 验证结果

本实验在 V7 版本的 FPGA 平台上进行验证。首先将启动的逻辑版本烧写至 FPGA 板中，然后经串口通过 Ymodem 协议把 BOOTROM 的镜像再烧写至 FPGA 板中。启动流程设计完毕会有如下的串口打印：

```
…
DEBUG Text:Bootrom Start!
DEBUG Text:Loader Branch!
DEBUG Text:launch_Loader
….
DEBUG Text:Key Copy Success
DEBUG Text:Verify Key Area:ROM Key
DEBUG Text:Verify Key Success
DEBUG Text:Code Copy Success
DEBUG Text:Decrypt Success
DEBUG Text:Verify Code With self Key Success
DEBUG Text:Version verify success
DEBUG Text:Verify Code Success
DEBUGText:Exiting main_entry:
```

其中含有 verify 的步骤都是使用 ECDSA 进行验证。首先头区代码验证成功，然后将代码段进行解密，最后代码段的数据和版本号验证成功。这些验证过程保证了 IOT 芯片启动流程的完整性和安全性。

综上，ECC 算法应用于 IOT 领域有如下优势：1）同等密钥长度下，ECC 比 RSA 安全强度更高；2）ECC 密钥长度较短，较短的密钥长度可降低资源占用。另外，对于 IOT 芯片而言，由于 ROM 比 RAM 价格更低，因此用安全驱动固化的设计方案可进一步降低成本。

结束语 本文针对物联网安全问题，概述了常用的公钥密码算法。通过对 RSA 和 ECC 的性能和安全强度的比较，优选 ECC 作为 IOT 应用的安全启动算法，并实现了应用于 IOT 芯片的安全驱动固化。众所周知，解决系统安全的问题必须从底层硬件结构开始，从整体上采取措施，才能保证系统的安全。安全驱动固化的设计从最底端出发，为系统启动过程提供度量依据，即首先确保 BOOTROM 引导并度量 BOOT 的启动过程可信，然后确保 BOOT 再去引导内核的过程的可信，这样一级度量一级从而可以保证系统整体的启动安全。不过，在验签速度方面，ECC 仍然慢于 RSA，后续工作需进一步对 ECC 算法进行优化，从而提升 IOT 芯片的启动效率。

参 考 文 献

[1] COLUMBUS L. 2018 roundup of Internet of Things forecast-sand marketestimates[EB/OL]. https://www.forbes.com/sites/louiscolumbus/2018/12/13/2018-roundup-of-internet-of-things-forecasts-and-market-estimates/#428d6ab27d83,2018.

[2] COLUMBUS L. Roundup of Internet of Things forecasts and market estimates[EB/OL]. https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#27232e3d292d,2016.

[3] ZHOU H L,LQUO X H,FENG X S. Thematic analysis on security issues of the Internet of Things [J]. Computer Knowledge and Technology,2019,15(12):17-18.

[4] BI R,LI X G,JIANG J. Internet of Things security incident cases and problem analysis [J]. Telecommunications Network Technology,2014(12):10-13.

[5] YAN Z,ZHANG P,VASILAKOS A V. A survey on trust management for Internet of Things[J]. Journal of Network & Computer Applications,2014,42(3):120-134.

[6] PETROULAKIS N E,TRAGOS E Z,FRAGKIADAKIS A G,et al. A lightweight framework for secure life-logging in smart environments[J]. Information Security Tech. Report, 2013,17(3):58-70.

[7] DI R,SALLERAS X,SIGNORINI M,et al. A blockchain-based Trust System for the Internet of Things[C]// ACM SACMAT 2018. ACM,2018.

[8] BYUNG-YOON S. A Public-key Cryptography Processor supporting P-224 ECC and 2048-bit RSA[J]. Journal of IKEEE, 2018,22(3):522-531.

[9] KOBLITZ N. Elliptic Curve Cryptosystem [J]. Mathematics of Computation,1987,47(177):203-209.

[10] BAO F,CHEN I R. Trust management for the internet of things and its application to service composition[C]// World of Wireless,Mobile & Multimedia Networks. IEEE,2012.

ZONG Si-jie, born in 1997, postgraduate. Her main research interests include embedded software and so on.

HE Long-bing, born in 1982, Ph.D, associate professor. His main research interests include microelectronics and solid-state electronics.