CRYPTO PRODUCTS

REAL.EASY TRAINING MANUAL

2nd QUARTER 2015

ATECC508A Crypto Element with ECDH Key Agreement

Table of Contents

1. *Two Families of Atmel Crypto Products* 6
2. *CryptoAuthentication Keeps it Real* 7
3. *SECURITY DRIVES GROWTH* 9
4. *WHY SHOULD CEOs CARE ABOUT SECURITY?* 10
5. *SECURITY MATTERS* 11
6. *EASY TO FIND UNPROTECTED NODES* 12
7. *VULNERABITY IS WIDESPREAD* 13
8. *HARDWARE IS BETTER…BUT IT MUST BE PROTECTED* 14
9. *SECURE HARDWARE IS THE BEST APPROACH* 15
10. *ATMEL MAKES IT EASY* 16
11. *HARDWARE KEY STORAGE BEATS SOFTWARE KEY STORAGE* 17
12. *ALWAYS ASK IF CRYPTO SHOULD BE IN ANY SYSTEM DESIGN***Error! Bookmark not defined.**
13. *Uses of Authentication* 19
14. *Why IoT (and everything else) Requires Strong Authentication* 20
15. *Why Buy CryptoAuthentication?* 23
16. *Real security is all about safe key storage.* 24
17. *Conditions today are creating a perfect storm for security* 25
18. *Use Cases:* Accessory Authentication 26
19. *Use Cases:* Medical Device Authentication 27
20. *Use Cases:* Firmware Authentication 28
21. *Use Cases:* Industrial Authentication 29
22. *Use Cases:* Consumable Authentication 30

23. *Use Cases:* Automotive Authentication.. 31
24. *Expanding Solution Coverage* 32
25. *Crypto Basics 1:* Symmetric and Asymmetric Authentication...................................... 33
26. *Crypto Basics 2:* Encryption and Authentication 34
27. *Crypto Basics 3:* Hashing vs. Encryption 35
28. *Crypto Basics 4:* Real Short Glossary... 36
29. *Crypto Basics 5:* “CIA” The Three Pillars of Security 37
30. *Symmetric Authentication* with secrets stored on the host using Challenge-Response.38
31. *Symmetric Authentication* without secret storage on the host using a Fixed Challenge39
32. *Symmetric Authentication* without secret storage on the host using Fixed Challenge and intermediate key. 40
33. *Asymmetric Authentication* using Digital Signatures (ECDSA) 41
34. *Asymmetric Authentication*: Making the ECDSA certificates...................................... 42
35. *Asymmetric Authentication*: Creating the Client’s Certificate 43
36. *Asymmetric Authentication*: Creating the Signer’s Certificate..................................... 44
37. *Asymmetric Authentication* ECDSA is a two phased process (Phase 1, Part 1: Verify Client’s Public Key) 45
38. *Asymmetric Authentication*: ECDSA is a two phased process (Phase 1, Part 2: Verify Issuer’s Signature) 46
39. *Asymmetric Authentication:* ECDSA is a two phased process (Phase 2: Verify Private Key) 47
40. *Firmware Protection:* Secure Boot (Step 1. Making the signature in the factory)......... 48
41. *Firmware Protection:* Secure Boot (Step 2. Authenticating the device in the field) 49
42. *Protecting Downloaded Firmware* with Symmetric Keys; Step1: Encrypt and MAC application code 50
43. *Protecting Downloaded Firmware* with Symmetric Keys; Step2: Download, Decrypt, and Authenticate 51
44. *Symmetric Session Key Exchange* (with crypto devices on both sides) 52
45. *Protecting Communication* between the Crypto device and MCU. 53
46. *Secure Storage* Using Encryption (Symmetric).. 54

47. Secure Password Protection 55
48. *Confidentiality* using Symmetric Session Key Exchange 56
49. *Node Data Integrity* Using Symmetric Intermediate Keys 57
50. *Node Authentication* Using Symmetric Intermediate Keys 58
51. *Confidentiality* using ECDH Key Agreement 59
52. *ATECC508A* provides Confidentiality, Integrity, and Authenticity. 60
53. Security and the IoT 61
54. *The Mathematical Magic* of ECDH Key Agreement 62
55. *Using diversified keys* to increase security 65
56. *Innovations* that make Atmel secure key storage devices robust. 66
57. Consumption Tracking 67
58. Secure Personalization 68
59. Secure Personalization Services 69
60. Additional Configuration Options 70
61. Atmel CryptoAuthentication ™ *Devices* 71
62. Competitive Advantages 72
63. AES encryption in the real world 73
64. Message Authentication Example 74
65. Intrusion Detection Feature (ATECC108A & ATECC508A) 75
66. *Feature Summary* of ATMEL ECC CryptoAuthentication devices (ATECC108A and ATECC508A) 76
67. *Feature Summary* of ATMEL ECC CryptoAuthentication devices (ATECC108A and ATECC508A) 77
68. Atmel CryptoAuthentication ™ *Devices: ATECC108A* 78
69. Atmel CryptoAuthentication ™ *Devices: ATSHA204A* 80
70. Atmel CryptoAuthentication ™ *Devices: ATAES132A* 82

71. *Atmel CryptoAuthentication ™ Devices: ATECC508A* 84

73. *Atmel CryptoAuthentication ™ ECC Device Comparison* 86

74. CryptoAuthentication Device Selector Guide 87

78. Atmel CryptoAuthentication Device Ordering Guide: ATECC108A.............................. 91

79. Atmel CryptoAuthentication Device Ordering Guide: ATSHA204A.............................. 92

80. Atmel CryptoAuthentication Device Ordering Guide: ATAES132A 93

81. Atmel CryptoAuthentication Device Ordering Guide: ATECC508A.............................. 94

82. *Atmel provides tools for every stage* 95

83. *AT88CK490 and AT88CK590 Demo-Evaluation kits* 96

84. *CryptoAuthXplained Pro Development Kit* 97

85. *AT88CK9000 Production Programming Tool* 98

86. *"ACES" (Atmel Crypto Evaluation Studio) Software*.. 99

87. Atmel CryptoAuthentication Kit Ordering Guide.. 100

88. *Atmel Trusted Platform Module (TPM)* 103

89. *Trusted Platform Module (TPM) Security Features*... 104

90. *Authentication Using TPM* 106

91. *Secure Boot Using TPM for Platform Integrity* 107

92. Atmel Trusted Platform Module (TPM) Device Selector Guide.................................. 108

95. Atmel Trusted Platform Module (TPM) Device Ordering Guide 111

97. *Trusted Platform Module (TPM) Kits*.. 113

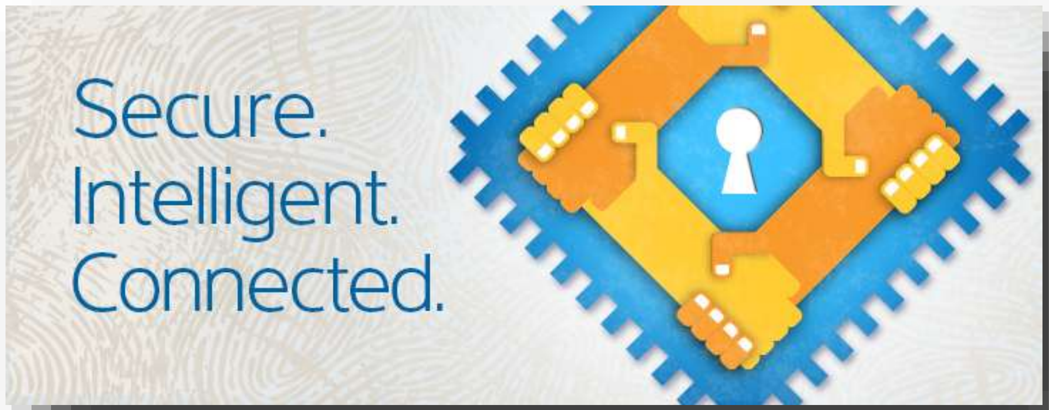
98. *Trusted Platform Module (TPM) Kits   Selector Guide* 114

99. *Contacts* **Error! Bookmark not defined.**

100. *FAQs:* Some important questions: 115

1. Two Families of Atmel Crypto Products

CryptoAuthentication ™ Device Family



Atmel offers the industry's widest portfolio of crypto elements with ultra-secure hardware-based key storage, to provide confidentiality, integrity, and authentication. There are now four industry leading integrated circuits with several others under development.

Trusted Platform Module (TPM) Family

Atmel Trusted Platform Module (TPM) devices employ ultra secure hardware-based key storage to implement public key (RSA) security for PCs, tablets, and embedded systems. They are complete turnkey systems on a single device integrating a microcontroller, EEPROM, and crypto engines, and are compliant to Trusted Computing Group TCG 1.2 specification Revision 116.

2. CryptoAuthentication KEEPS IT REAL

Atmel's crypto elements wity hardware-based key storage ensure that a product, consumables it uses, firmware it runs, accessories that support it and, the network nodes it connects to are not cloned, counterfeited, or tampered with.

Using cryptographic procedures and advanced defense mechanisms the devices provide strong authentication (i.e. "keep it real").

In addition to authetnication, CryptoAuthentication devices also make it easy to add the other foundational pillars of security, namely confidentiality and data integrity, to microprocessor-based systems.

3. *SECURITY DRIVES GROWTH*

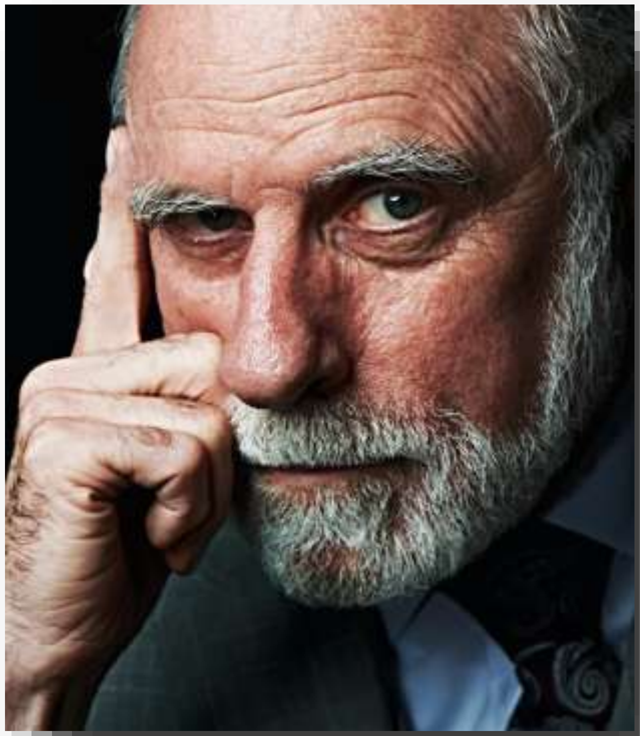
IoT, cloud, wearables, vehicle–to-vehicle communications, and mobile market growth will give rise to billions of smart nodes and platforms.

That multiplies the number of entry points hackers can attack.

So, robust security is needed.

The Internet's inventor, Dr. Vint Cerf, says that strong authentication is important to make sure that the IoT is talking to the devices they are supposed to talk to.

We completely agree.



4. *WHY SHOULD CEOs CARE ABOUT SECURITY?*

CEOs will face shareholder and class action lawsuits from data breaches.

Proven means to secure against hacking and cloning exist, there is a perceived duty to employ them, and not doing so can be argued as being negligent.

Agree or not, such arguments are already starting to happen.

Some CEOs and CTOs are mandating use of strong hardware-based security to protect their companies and get a competitive edge.

We can see why.

5. SECURITY MATTERS MORE THAN EVER

Even with the explosion of data breaches, security is still treated as an afterthought.

Security should, in fact, be the prerequisite of any discussion about a data system or any type.

Engineers, executives, investors, and researchers alike have been whistling past the graveyard hoping that their digital interests will not be attacked too badly.

Of course that is irrational because targets are super easy to find now.

6. *IT IS EASY TO FIND UNPROTECTED NODES*

Traffic lights, security cameras, home automation devices, appliances, heating systems, industrial networks…you name it… are now super easy to find because of Shodan, which is the Google of embedded systems.

Easy to find unprotected nodes!
SHODAN
Search
EXPOSE ONLINE DEVICES.
WEBCAMS. ROUTERS.
POWER PLANTS. IPHONES. WIND TURBINES.
REFRIGERATORS. VOIP PHONES.
TAKE A TOUR
FREE SIGN UP

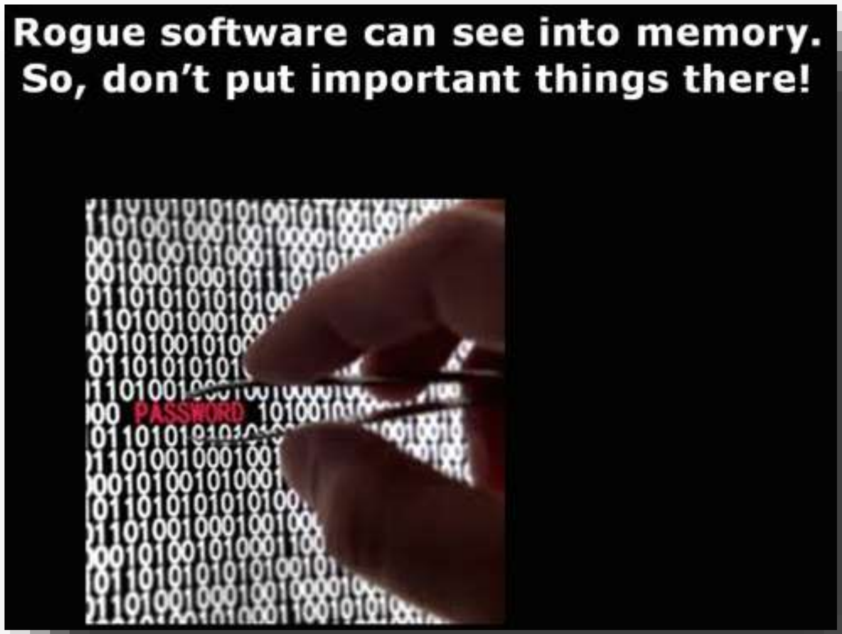
Shodan searches the Internet's back channels 24/7 looking for servers, webcams, printers, routers and all the other stuff connected to the Internet.

Shodan alone proves that robust security for connected devices is needed badly because if it is connected, it is vulnerable to hackers.

7. *VULNERABITY IS WIDESPREAD*

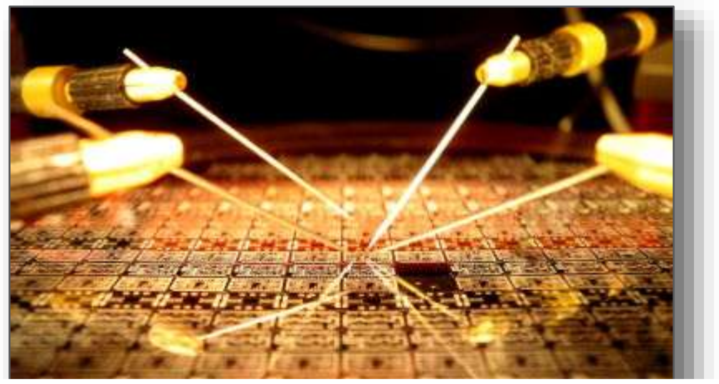
Hackers steal passwords, digital IDs, IP, and financial data because software is used to protect system software.

But, all software is vulnerable because all software has bugs.

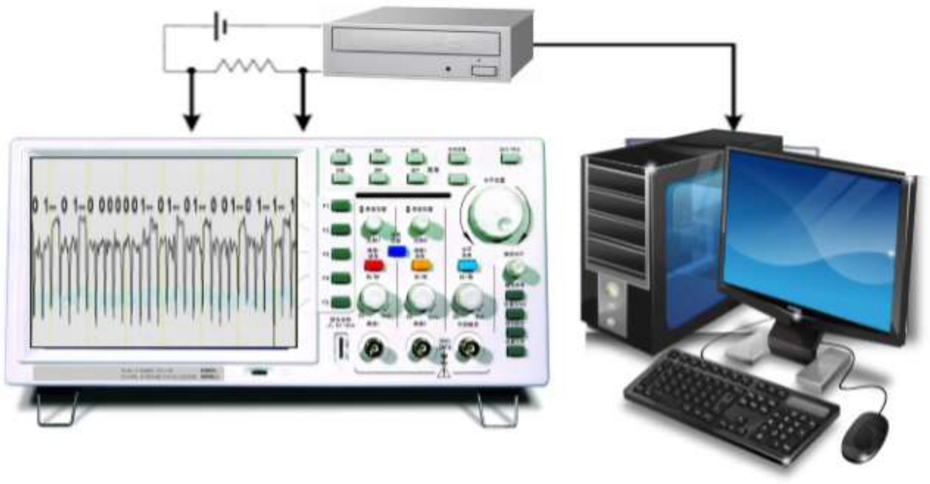

Rogue software can see into memory.
So, don't put important things there!
PASSWORD

8. *HARDWARE IS BETTER…BUT IT MUST BE PROTECTED*

Hardware is better, but integrated circuits can be probed to read what is on the circuit.



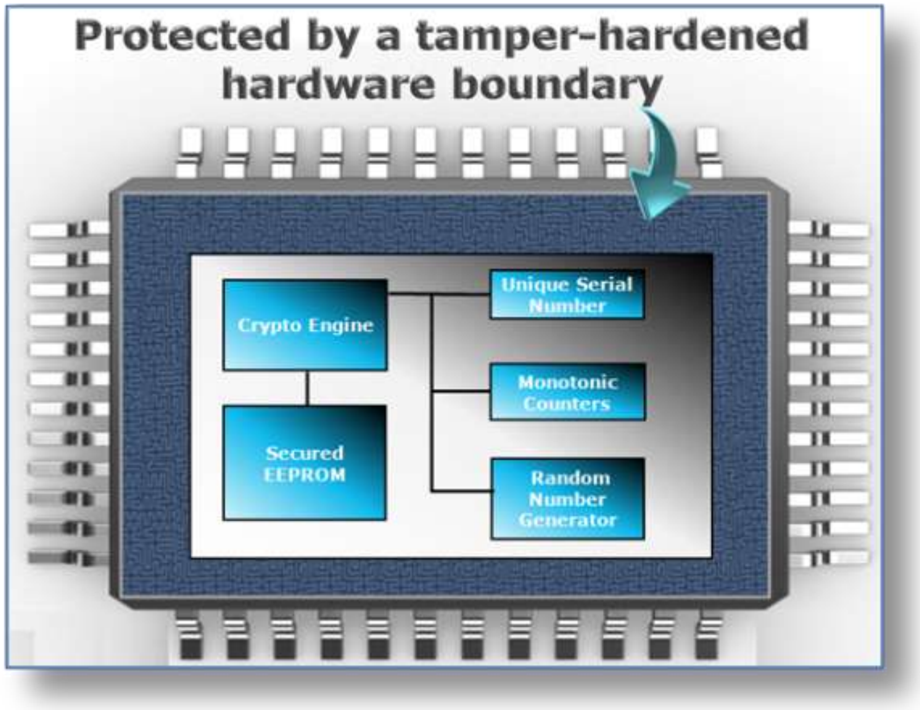
Also, power analysis can extract secrets.



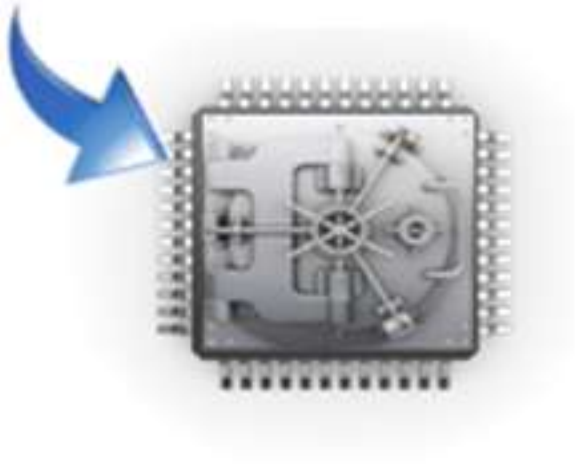
So, a more secure approach is needed.

9. *SECURE HARDWARE IS THE BEST APPROACH*

Hardware-based key storage with physical barriers and cryptographic countermeasures can fight off even the most aggressive attacks.



Once keys are locked away in protected hardware, attackers cannot see them and cannot attack.



10. *ATMEL MAKES IT EASY*

CryptoAuthentication™ and Trusted Platform Module (TPM) devices are turn-key, cost effective, and work with any MCU!

Atmel does the hard cryptographic engineering, so users don’t need to be crypto experts.

That makes it REAL EASY to add robust security to any system !

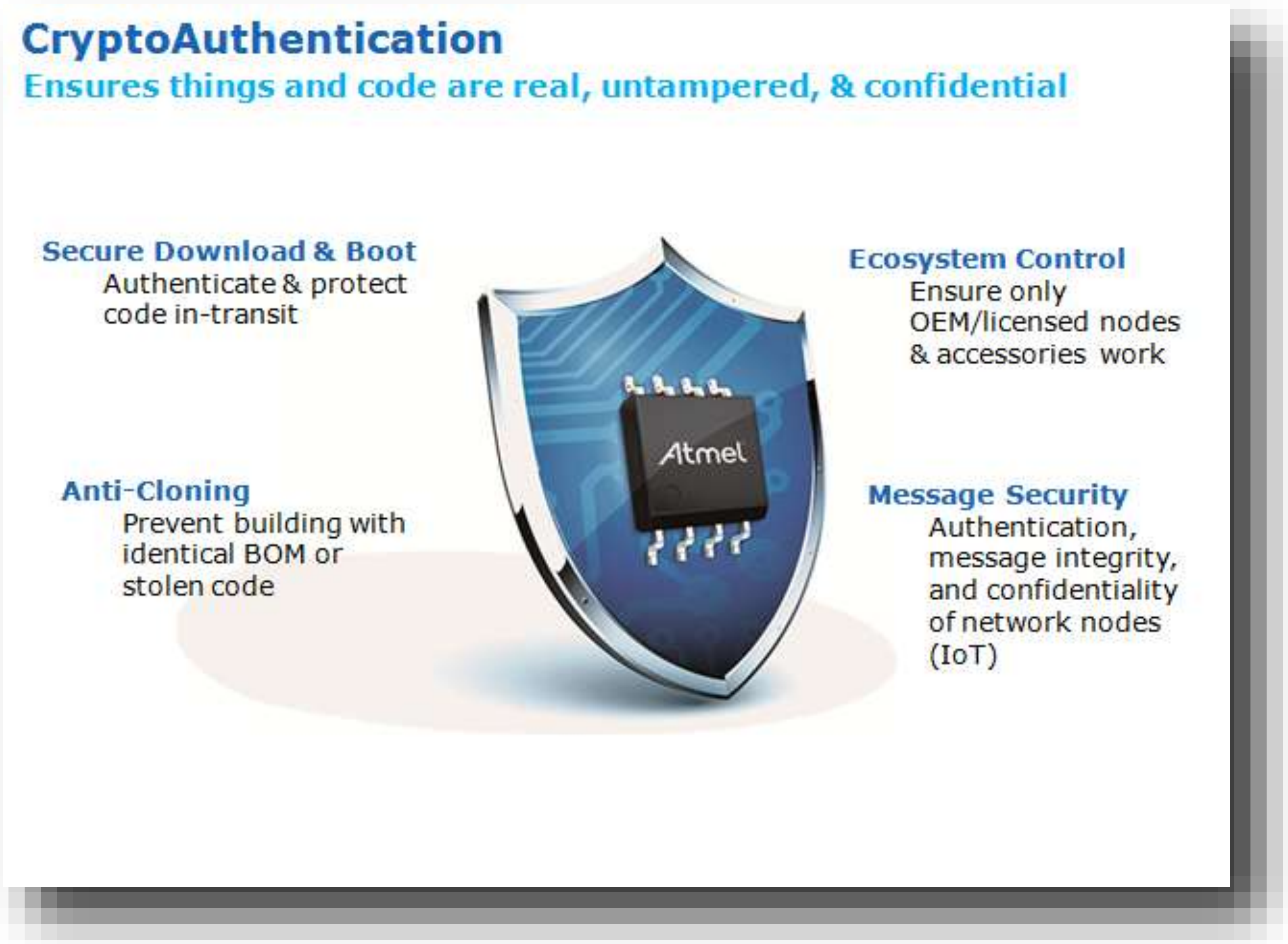
11. *HARDWARE KEY STORAGE BEATS SOFTWARE KEY STORAGE*

An important (Fortune 100) industrial networking company executive said that not using hardware key storage is like storing your cryptographic key in a wet paper bag.

He totally gets it.

What about you?

12.



CryptoAuthentication

Ensures things and code are real, untampered, & confidential

Secure Download & Boot
Authenticate & protect code in-transit

Anti-Cloning
Prevent building with identical BOM or stolen code

Ecosystem Control
Ensure only OEM/licensed nodes & accessories work

Message Security
Authentication, message integrity, and confidentiality of network nodes (IoT)

13. Uses of Authentication

- **Anti-Counterfeiting**

  Validates that a removable, replaceable, or consumable client is authentic. Examples of clients could be system accessories, electronic daughter cards, or other spare parts. It can also be used to validate a software/firmware module or memory storage element.

- **Protecting Firmware or Media**

  Validates code stored in flash memory at boot to prevent unauthorized modifications, encrypt downloaded program files as a common broadcast, or uniquely encrypt code images to be usable on a single system only.

- **Storing Secure Data**

  Store secret keys for use by crypto accelerators in standard microprocessors. The ATECC508A can also be used to store small quantities of data necessary for configuration, calibration, ePurse value, consumption data, or other secrets. Programmable protection is available using encrypted/authenticated reads and writes.

- **Checking User Passwords**

  Validates user-entered passwords without letting the expected value become known, maps memorable passwords to a random number, and securely exchanges password values with remote systems.

Benefits

- **Preserves revenue streams from consumables**
- **Protects Intellectual Property**
- **Keeps data secure**
- **Restricts unauthorized access**

14. *Why IoT (and everything else) Requires Strong Authentication*

It seems that every day new and increasingly dangerous viruses are infecting digital systems. Viruses with names such as Heartbleed, Shellshock, Poodle, and Bad USB have put innocent people at risk in 2014 alone. Russian Cyber gangs (a.k.a. “CyberVor”) have exposed over a billion passwords. The scary thing is that the attacks are targeted at the very security mechanisms that are meant to provide protection. Because the digital protection mechanisms themselves have become targets, they must be hardened. This is especially important now that the digital universe is going through a type of Big Bang with the explosion of the IoT. That is sending billions of little sensing and communicating processors all over the earth, like dust. Growth in processing, communicating, and sensing semiconductors (which are exactly what the IoT is made from) will grow at a rate of over 36% in 2015 according to Gartner, dwarfing the overall semiconductor market growth of 5.7%. Big Bang. Big Growth.

As for security, the IoT will multiply the number of points for infection that hackers can attack by many orders of magnitude.

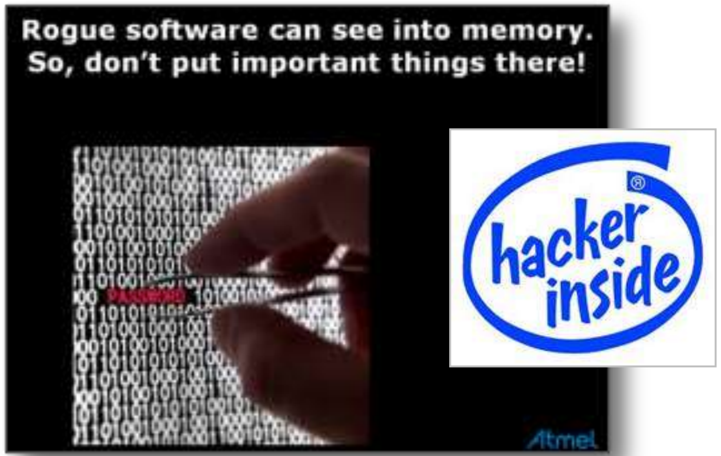
It is not hard to see that trust in the data communicated via an ubiquitous (and nosey) IoT will be necessary for the IoT to be widely adopted. Without trust, the IoT will fail to launch. There is hardly any doubt there. In fact, the recognized inventor of the Internet, Vint Cerf, completely agrees, saying that the for IoT strong authentication is important, and we need to make sure they are talking to the devices they are supposed to talk to.. Those are very clear statements, but for fun let’s translate Dr. Cerf’s admonition into pop culture parlance: “No security? No IoT for you.”

Why IoT (and everything else) Requires Strong Authentication (Continued)

Lawsuits. But, there is much more to the story behind why the IoT needs strong security. Because the world has become hyper-connected, financial and other sensitive transactions have become almost exclusively electronic. Money now is simply electronic data, so everyone and every company are at risk of financial losses stemming directly from data breaches. See? Databanks are where the money is kept and data is what criminals attack. While breaches are in fact being publicized, there has not been much open talk about their leading to significant corporate financial liability. That liability, however, is real and growing. So, CEOs should not be the least bit surprised when they start to be challenged by significant shareholder and class action lawsuits stemming from security breaches.

Although inadvertent, companies are in fact exposing identities and sensitive financial information of millions of customers, and they may not always be taking all the measures that they can to ensure the security and safety of their products, data, and systems. Both exposure of personal data and risk of product cloning can translate to financial damages. Damages translate to legal action. The logic of tort and securities lawyers is that if proven methods to secure against hacking and cloning already exist, then it is the fiduciary duty of the leaders of corporations (i.e. the C-Suite occupants) to embrace such protection mechanisms (such as hardware-based key storage), and not doing so could possibly be argued as being negligent. Agree or not, that line of argumentation is logical and perhaps likely.

A few CEOs have already started to equip their systems and products with strong hardware-based security devices...but they are doing it quietly and not telling their competitors. This gives them an edge.


Rogue software can see into memory.
So, don't put important things there!
PASSWORD
hacker inside
Atmel

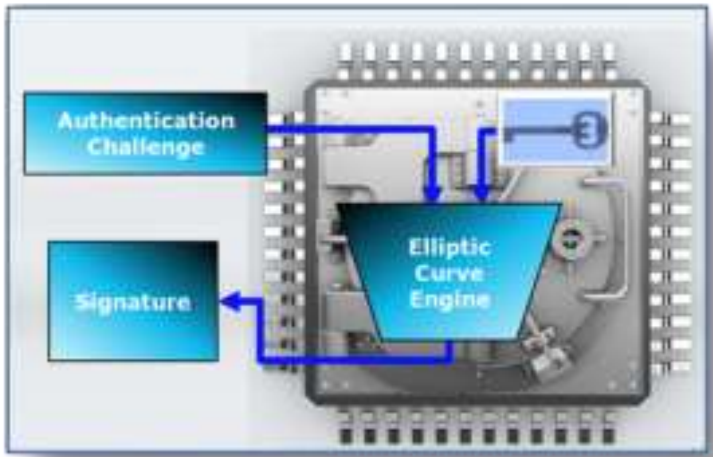
Software, Hardware, and Hackers. Why is it that hackers are able to penetrate systems and steal passwords, digital IDs, intellectual property, financial data, and other secrets? It is because until now only software has been used to protect software from hackers. Hackers love software. Breaking into software is what they live for.

The problem is that rogue software can see into system memory, so it is not a great place to store important things such as passwords, digital IDs, security keys, and other valuable things. The bottom line is that all software is vulnerable because software has bugs despite the best efforts of developers to eliminate them. So what about storing important things in hardware?

Why IoT (and everything else) Requires Strong Authentication (Continued)

Hardware is better, but standard integrated circuits can be physically probed to read what is on the circuit. Also, power analysis can quickly extract secrets from hardware.   So, is there anything that can be done? The answer fortunately is yes.

Three of four generations of hardware key storage devices have been designed to protect keys with physical barriers and cryptographic countermeasures that ward off even the most aggressive attacks. Once keys are securely locked away in protected hardware attackers cannot see them and they cannot attack what they cannot see.   Secure hardware key storage devices like Atmel CryptoAuthentication™ employ both cryptographic algorithms and a tamper-hardened hardware boundary to keep attackers from getting at the cryptographic keys and other sensitive data.

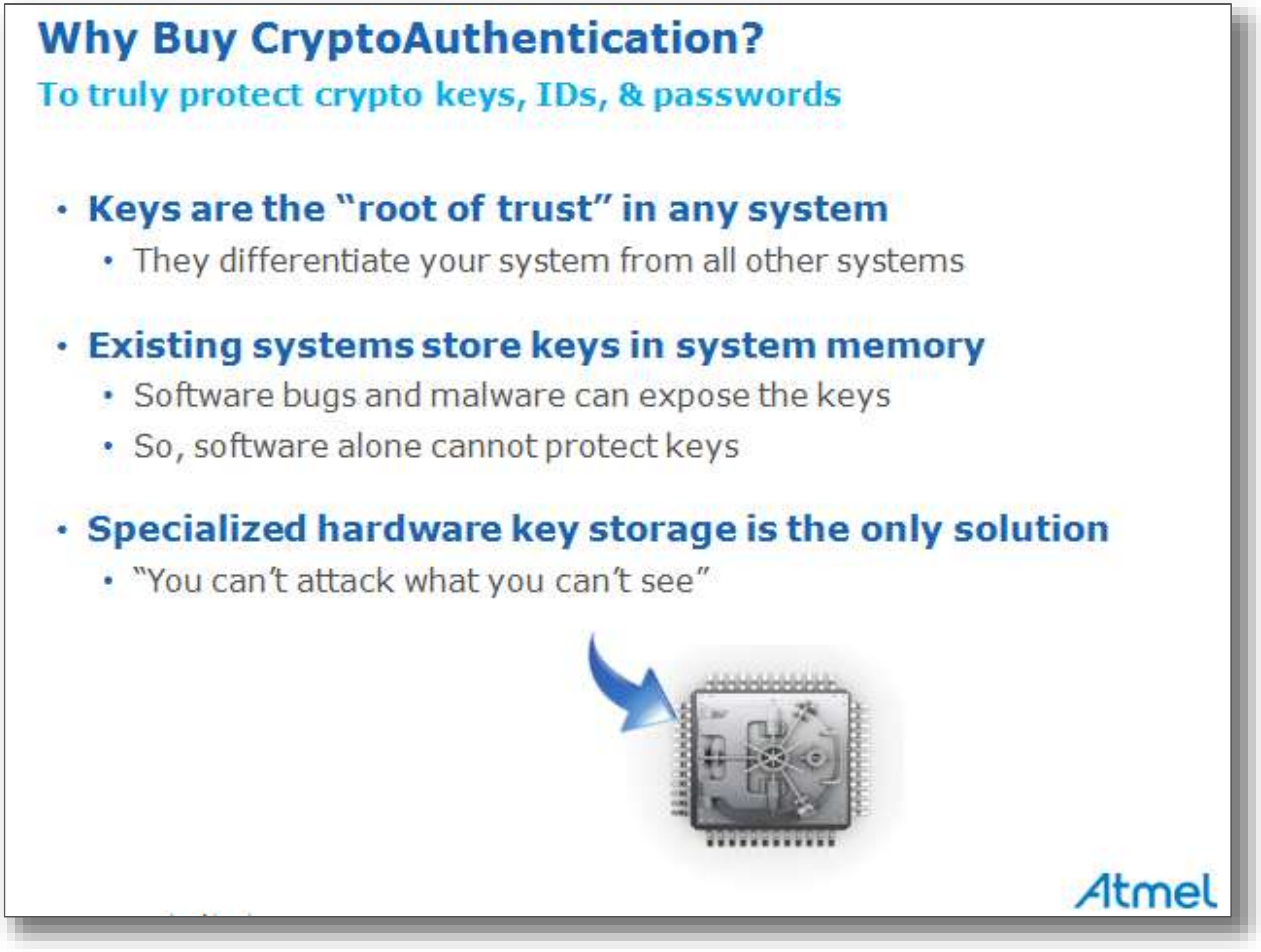


Keeping secrets keys secret. The basic idea behind such protection is that cryptographic security depends on how securely the cryptographic keys are stored. But of course it is of no use if the keys are simply locked away. There needs to be a mechanism to use the keys without exposing them.   That is the other part of the CryptoAuthentication™ equation, namely built-in crypto engines that run cryptographic processes and algorithms. A simple example of accessing the secret key without exposing it is using challenges (usually random numbers), secret keys, and cryptographic algorithms to create unique and irreversible signatures that provide security without anyone seeing the secret key.

Crypto engines make running complex mathematical functions easy while at the same time keeping secret keys secret inside robust, protected hardware.   This hardware key storage/crypto engine combination is the secret to keeping secrets and being easy to use, available, ultra-secure, tiny, and inexpensive

While the engineering that goes into hardware-based security is sophisticated, **Atmel does all the crypto engineering so there is no need to become a crypto expert.**

15. Why Buy CryptoAuthentication?



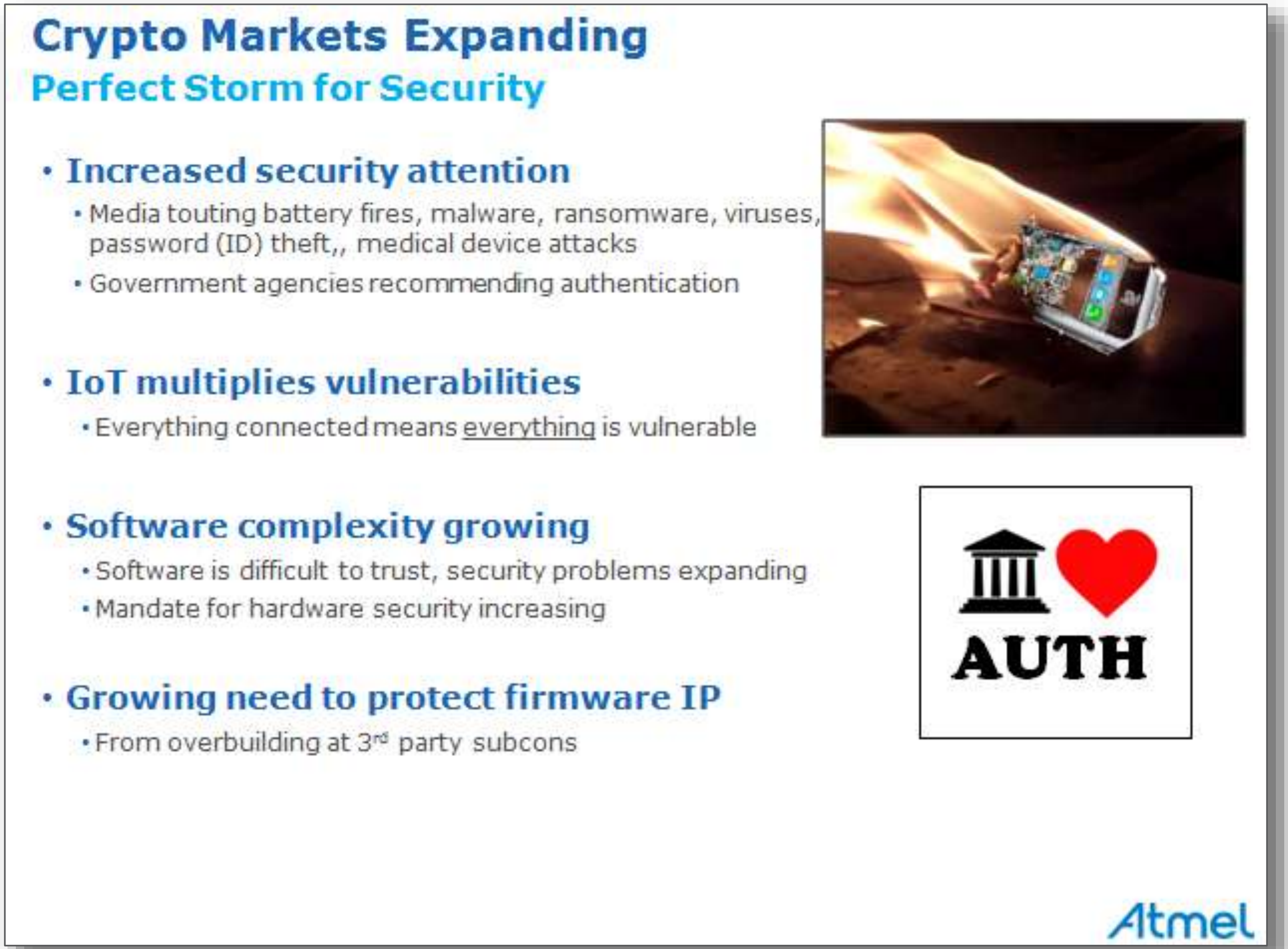
Why Buy CryptoAuthentication?

To truly protect crypto keys, IDs, & passwords

- **Keys are the "root of trust" in any system**
  - They differentiate your system from all other systems
- **Existing systems store keys in system memory**
  - Software bugs and malware can expose the keys
  - So, software alone cannot protect keys
- **Specialized hardware key storage is the only solution**
  - "You can't attack what you can't see"

16. *Real security is all about safe key storage.*

17. *Conditions today are creating a perfect storm for security*



Crypto Markets Expanding

Perfect Storm for Security

- **Increased security attention**
  - Media touting battery fires, malware, ransomware, viruses, password (ID) theft,, medical device attacks
  - Government agencies recommending authentication
- **IoT multiplies vulnerabilities**
  - Everything connected means everything is vulnerable
- **Software complexity growing**
  - Software is difficult to trust, security problems expanding
  - Mandate for hardware security increasing
- **Growing need to protect firmware IP**
  - From overbuilding at 3rd party subcons

Atmel

18. *Use Cases:* Accessory Authentication

One of today's main authentication use cases are accessories and consumables. Consumables are sort of like accessories that are bought, used up, disposed of, and re-bought, again and again. There is a wide spectrum of accessories, particularly for mobile handsets, already in the market with new ones being introduced all the time. Examples are wired and wireless battery chargers, docking stations, speakers, wearables, wireless camera lenses, smart covers, advanced earphones, headphones, and medical diagnostic sensors. Accessories and consumables present very similar business models regarding authentication, but consumables will drive higher volumes due to emerging products.

Atmel
ATSHA204

The constellation of accessories that surround mobile phones, tablets, computers, GPS units, digital cameras, in-car entertainment equipment, and other platforms represents a very large market. According to market research, by 2017 mobile phone and tablet accessory revenues could reach over $75 billion. Such a sizeable forecast is generating huge incentives for both bona fide competitors and illegitimate suppliers to jump in. Hardware authentication is the obvious solution to protect market share. Safety is another critical issue. There are many recent instances of batteries exploding or catching fire in mobile and other products. A bad accessory experience like a fire reflects not only on the accessory maker but the main platform brand even more so. Authentication makes certain that only safe batteries can be used, which is great for consumers and the legitimate suppliers.

Customers also support this model as the desirability of authentic, uniquely marketed accessories has gained popularity in recent years. Consumers made weary by cloned products not performing to their expectations are becoming more and more willing to pay the extra cost of a well designed accessory. CryptoAuthentication is ideally suited for this market, and is enabling manufactures to support customer trends while providing an exceptional tool to manage their own product lines. In short: Authentication enhances revenue, brand equity, and safety. And that makes a real difference in the real world.

Benefits

- **Protects revenue stream**
- **Protects brand image**
- **Allows manufacturers to control third party licenses**
- **Better control of the supply channel**
- **Enhanced product safety**

19. *Use Cases:* Medical Device Authentication

Authentication of medical devices is a growing market and it has started to be driven by government agencies. Governments have become the main drivers of medical policies and are clearly shaping the future of medical care. New policies will have enormous impact on the design, distribution, financing, and use of medical equipment of all kinds. Looking at where the US government, for example, wants medicine to go we can see two clearly emerging vectors; namely electronic records and telemedicine. Both of these vectors point in the direction of authentication.
Handset/tablet based telemedicine using an array of wired and wireless diagnostic sensors will help extend healthcare services to underserved and remote populations. Implantable sensors and devices are already being programmed wirelessly such as pacemakers and defibrillators, and the FDA has expressed concern that such products could be interfered with or hacked with mal-intent. So, in mid-2013 the FDA issued guidelines for authentication of wireless medical appliances in order to promote safety. Without a doubt, the last thing a patient with a pacemaker wants to have to worry about is someone hacking their heartbeat.

Authentication is also a way to ensure privacy of electronic medical records, which is required by laws such as HIPAA. The mandate for electronic records went into effect in January 2014. The need for authentication of electronic records will only grow as the systems continue to roll out and evolve.

Benefits

- **Secures ecosystems**
- **Improves quality**
- **Tracks charges**
- **Reduces medical liability exposure**

20. *Use Cases:* Firmware Authentication

Firmware in every type of product is vulnerable to cloning. Using CryptoAuthentication devices can ensure that only the authorized firmware is loaded at boot time. There is a growing trend in many industries where Intellectual Property (IP), particularly firmware is being copied, and the benefits of expensive R&D investments end up being forfeited by the legitimate owners. CryptoAuthentication is a simple and effective way to guard against firmware and product cloning.

Placing a CryptoAuthentication device onboard a micro-controlled system provides a simple solution that can match the secret stored in the security IC to the operating firmware. How that can be accomplished depicted in the secure boot diagrams (shown later) showing step one where the application code is signed in the factory and step two where it is verified by the Crypto device before launching the application.

Benefits

- **Prevents cloning**
- **Protects investments in firmware**

21. *Use Cases:* Industrial Authentication

Industrial Authentication
Protects industrial accessories & network nodes
Atmel

Authentication in the industrial space is very much like the consumer accessories and consumables segment. An embedded system in an industrial setting will also have various "things" that it connects to and uses.

Examples are sensors, firmware, chucks, tooling, counters, actuators, motors, fittings, control panels, power sources, batteries, spare parts, bits, dies, blades, materials, safety items, chemicals, authorized users, access points, I/O blocks, conveyors, valves, illumination…well, you get the picture.

The point is that industrial applications are not only very technical but widely variable, which means that there are numerous opportunities for adding authentication devices in industrial settings. Doing so enhances safety, tracks items, fights cloning, protects firmware, and ensures the source of spare parts, among other things.

Benefits

- **Prevents cloning**
- **Protects investments in firmware**
- **Enhances safety**
- **Tracks items**

22. *Use Cases:* Consumable Authentication

Finished crypto device is sent for assembly into a customer's product (such as an accessory) that will later be authenticated in the field
Atmel
Atmel
Accessories, Ltd.
Atmel
Filter
BOSCH

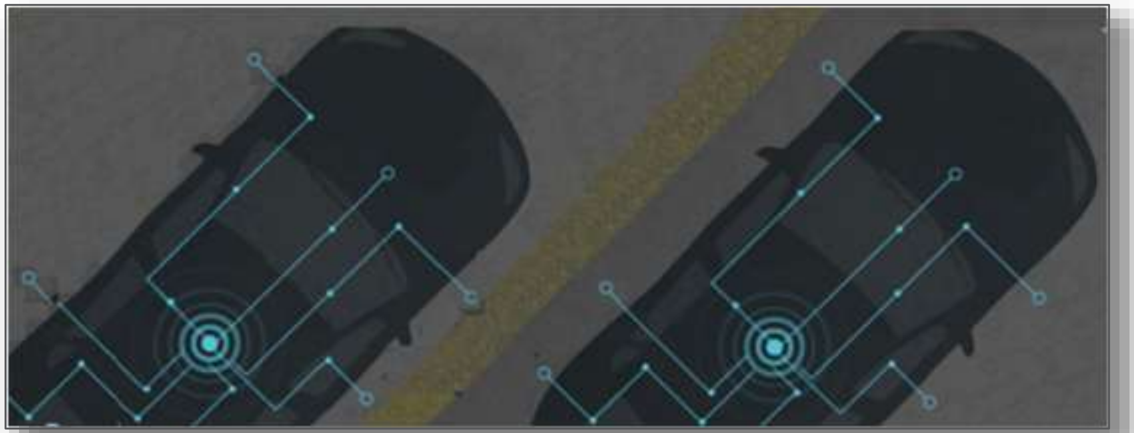
The food chain for crypto devices used in a consumable product is illustrated in the picture. Finished crypto devices are shipped from the factory with the customer's programming sealed in. This programming includes the stored secrets and operational programming. The customer's programming is also called "personalization" because it is specific to that customer (i.e. it is configured according to their design) . The crypto device is delivered to the customer's factory or assembly facility and either soldered to a PCB or glued to the surface of a product using the 3-contact RBH package. The PCB or the device itself (in the case of the RBH package) is attached inside or to the surface of the consumable). That consumable product then gets inserted into the host system, which can be a printer, refrigerator, medical device, or any number of products. That product when exhausted gets replaced in the field. When the consumable is put into the host system the authentication process gets initiated and lets the host know if that consumable is real or not.

Benefits

- **Prevents cloning**
- **Protects investments in firmware**
- **Enhances safety**
- **Protects revenue stream**
- **Protects brand image**
- **Better control of the supply channel**

23. *Use Cases:* Automotive Authentication

With Vehicle-to-Vehicle (V2V) and Vehicle to Internet (V2I) communications cars will “talk and listen” to one another — automatically. They will share information like proximity, speed, direction, road conditions, and sense other cars, motorcycles, and pedestrians. The chief driver of V2V is signaling impending collisions so that the cars can automatically take countermeasures.



While it may seem revolutionary, V2V is really a branch of Internet of Things (IoT). Describing IoT and V2V as equations, they could be expressed in the following way:

IoT = (MCU + Sensor + Security + Wireless) Low Power

V2V = IoT + Car

Equation two states that V2V is really the IoT on wheels. However, this spread of electronic devices presents a huge challenge, which is safety. Safety can be compromised by hackers interfering with V2V communications and hacking into automotive control systems. Safety and security are closely related. Clearly there needs to be strong security on ALL systems in cars. The US Department and Transportation (DoT) and National Highway Traffic Safety Administration (NHTSA) are partnering with research institutions and auto companies to collaborate on technology development and interoperability of V2V to promote traffic safety. V2V can transform the automotive experience. The danger is that remote communication opens the door for hackers who want to intercept, spoof, and misuse data. So, strong security becomes the final piece of the picture, and arguably the element that makes IoT/V2V even possible to be widely adopted. Of course the strongest form of security comes from hardware-based key storage.

24. *Expanding Solution Coverage*

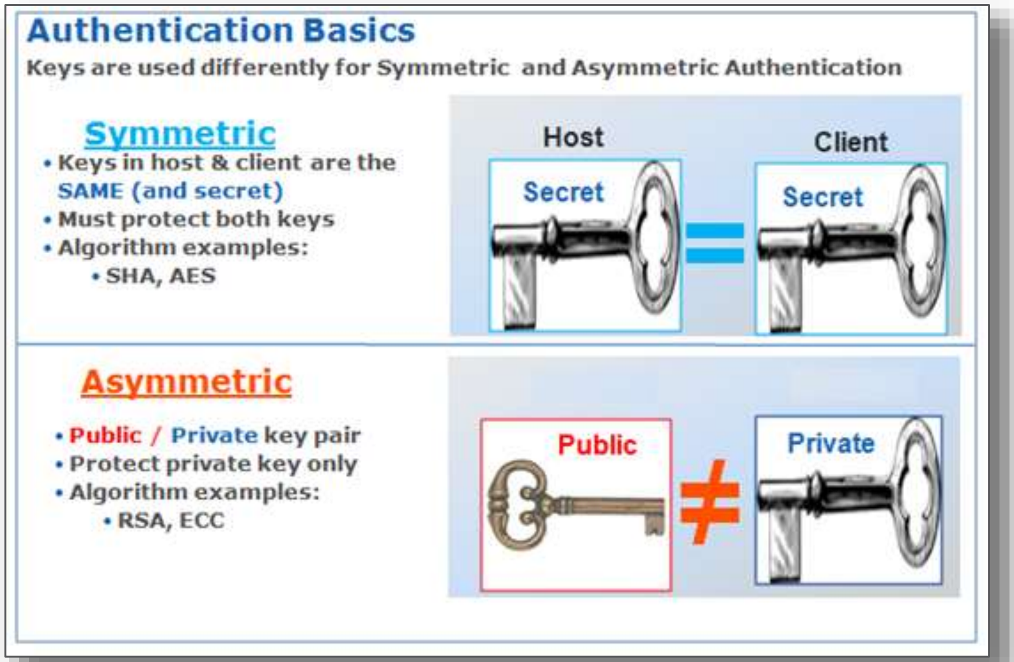
CryptoAuthentication devices cover parent-child and peer-to-peer application platforms. This means that there is a solution for all types of host-client and peer to peer relationships. Typically authentication has been parent to child, but with the IoT and car to car communication that is changing very rapidly. IoT, for example will have both hast-client and peer-to-peer requirements. The flexibility of the CryptoAuthentication portfolio presents valuable flexible options to designers.

Benefits

- **Covers range of parent-child and peer-to-peer solutions**
- **Perfect fit for new applications, such as Car 2 Car, smart covers, mobile batteries, IoT, etc.**
- **Ideal way to comply with evolving governmental regulations driving such as medical authentication and V2V**

Expanding Solutions
Atmel Covers Parent, Child, & Peer Platforms
75
Parent-Child
Peer-to-Peer

25. *Crypto Basics 1:* **Symmetric and Asymmetric Authentication**


Authentication Basics
Keys are used differently for Symmetric and Asymmetric Authentication
Symmetric
• Keys in host & client are the SAME (and secret)
• Must protect both keys
• Algorithm examples:
• SHA, AES
Host
Client
Secret
Secret
Asymmetric
• Public / Private key pair
• Protect private key only
• Algorithm examples:
• RSA, ECC
Public
Private

Authentication can be accomplished in two main ways: Symmetric and Asymmetric. The main difference between the two is related to the use of keys. If the same secret key is used on the client and also on the host then the application is symmetric, just like the name indicates. Remember if the keys are equal then the system is symmetric. Alternatively, if there is a mathematically related private and public key pair being used then the application is asymmetric. If the keys are not the same on each side then it is asymmetric. Atmel has devices for both types.

Asymmetric, asymmetric is also called public-key infrastructure or "PKI" and it is very well suited for real-world use. In fact, the internet is a big user of PKI. Note that Public Keys indicate that the system will be Asymmetric. Like is sounds, a public key is available to anyone. Think of a phone book filled with keys when envisioning the public key. Anyone having the public key can send encrypted messages to the owner of the private key. When a receiver gets an asymmetric message, he or she will decrypt it with their private key. In contrast, because symmetric cryptography uses the exact same key for both encryption and decryption, only the senders and receivers with that specific private key can communicate to each other. In addition to such differences in who can send and receive, other trade-offs between symmetric and asymmetric apply. For example, for symmetric at least twice the number of keys must be protected, increasing security risk. On the other hand, Asymmetric algorithms require more processing and thus are slower than symmetric. So, sometimes a combination of both symmetric and asymmetric is used to apply the relative advantages of each: the speed of symmetric and the security of asymmetric.

Benefits

- Verifies the identity of the sender and the integrity of the data
- Symmetric authentication can be fast
- Asymmetric authentication does not need secret storage in the host

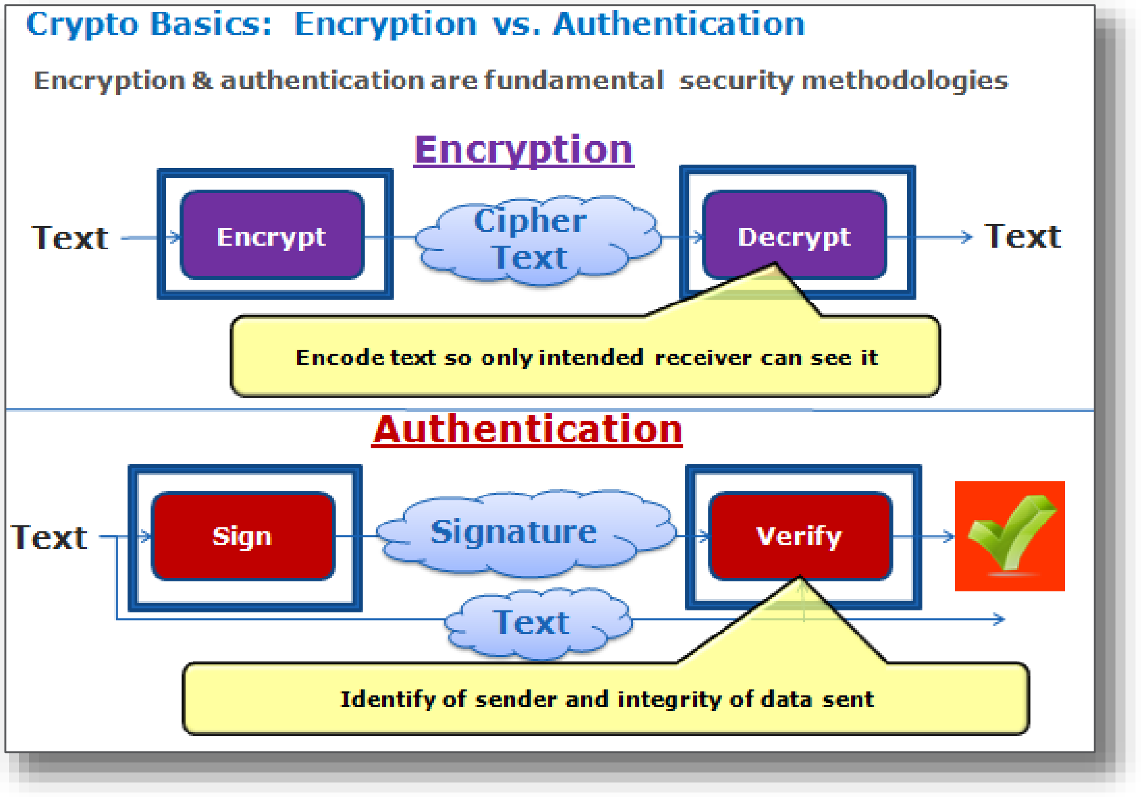
26. *Crypto Basics 2:* Encryption and Authentication

Encryption and authentication are fundamental security functions, but have different purposes.

Encryption, from the time of Julius Caesar until now, has been used to protect messages from being read by unintended people.

Authentication is the other main pillar of cryptography, and just like its name implies, authentication is about making sure that something is real.

In the cryptography world authentication is used to see if a message is real. To make such a "reality check" a number of things have to be verified such as if it was sent by the right sender, if messages were received in the right order, if the intended message or part of the message was deleted, or if a message was altered in some way. It doesn't matter if an encrypted message is decrypted accurately if it was not the intended message to begin with, or if it was altered. In short, encryption is about encoding and decoding, while authentication is about verifying the identity of the sender and the integrity of the message.



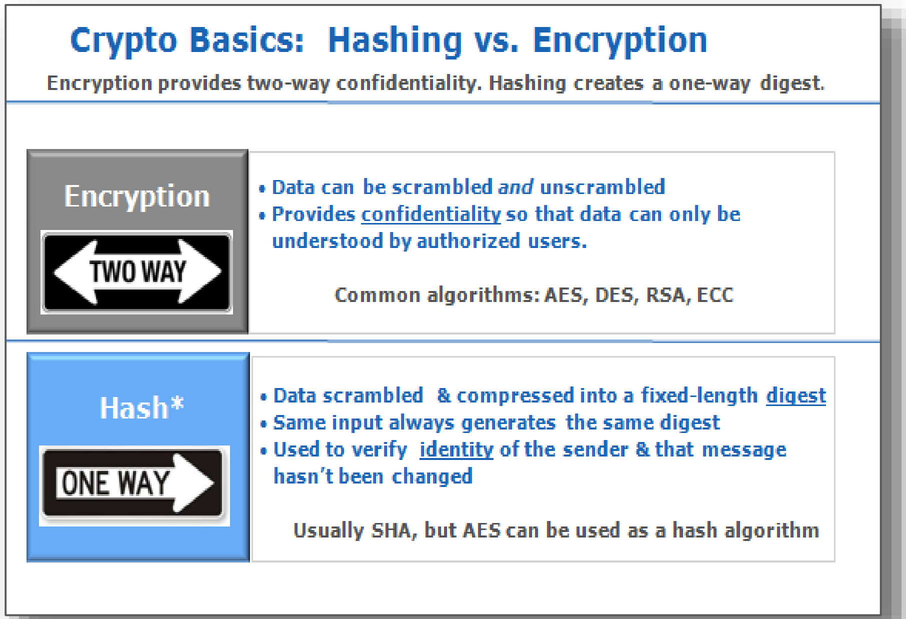
Crypto Basics: Encryption vs. Authentication
Encryption & authentication are fundamental security methodologies
Encryption
Text
Encrypt
Cipher Text
Decrypt
Text
Encode text so only intended receiver can see it
Authentication
Text
Sign
Signature
Verify
Text
Identify of sender and integrity of data sent

Benefits

- **Encryption keeps the message secret so only authorized receiver can see it.**
- **Authentication ensures the identity of the sender and integrity of the data**
- **Authentication and encryption can be used together for more security**

27. *Crypto Basics 3:* Hashing vs. Encryption

One of the basic tools used in authentication is a mathematical operation called a "hash function".



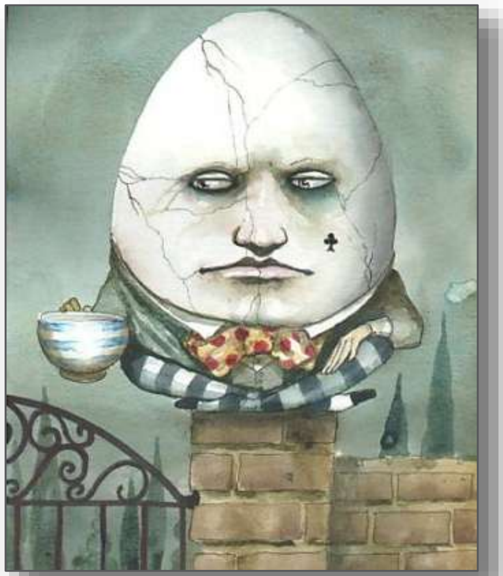
Crypto Basics: Hashing vs. Encryption

Encryption provides two-way confidentiality. Hashing creates a one-way digest.

Encryption (TWO WAY)	• Data can be scrambled *and* unscrambled • Provides confidentiality so that data can only be understood by authorized users. Common algorithms: AES, DES, RSA, ECC
Hash* (ONE WAY)	• Data scrambled & compressed into a fixed-length digest • Same input always generates the same digest • Used to verify identity of the sender & that message hasn't been changed Usually SHA, but AES can be used as a hash algorithm

- **A hash function** is defined as a group of characters that is mathematically mapped to a value of a certain length (called a hash value, hash, compression, fingerprint, message digest, or simply digest).
- **The hash value** is representative of the original string of characters, but is normally smaller than the original

Hashing functions have been used by computers for a long time and they are fundamental to cryptography. The hash value is representative of the original string of characters, but is normally smaller than the original. A hash is a one-way operation. The "one-wayness" of a hash function is its most important feature for cryptography because it is mathematically infeasible to reverse the hashing process to obtain the original message. A way to look at is that once the message is compressed it is impossible to uncompress it. Sort of like Humpty-Dumpty, you can't put it back together again. Also, a feature of a hash function is that any change to the input changes the digest, so hashes are great to create digital signatures that identify and authenticate the sender and message. Hashes are also used for secure password storage, file identification, message authentication coding (MAC), and asymmetric sign verify operations. Encryption is a different process from hashing and has a separate purpose. With encryption, data is scrambled and unscrambled in such a way that the input and output mapping is always one-to-one for a given key and is unintelligible to anyone other than a receiver who has the key to unscramble it. Encryption is always reversible (by definition), so encryption is used whenever there is the need to get the input data back out. However, the identity of the sender and the integrity of the message (meaning if it has been altered or not) are not guaranteed by encryption only. That is where authentication comes in. Authentication and encryption are two distinct and critical parts of strong cryptography.



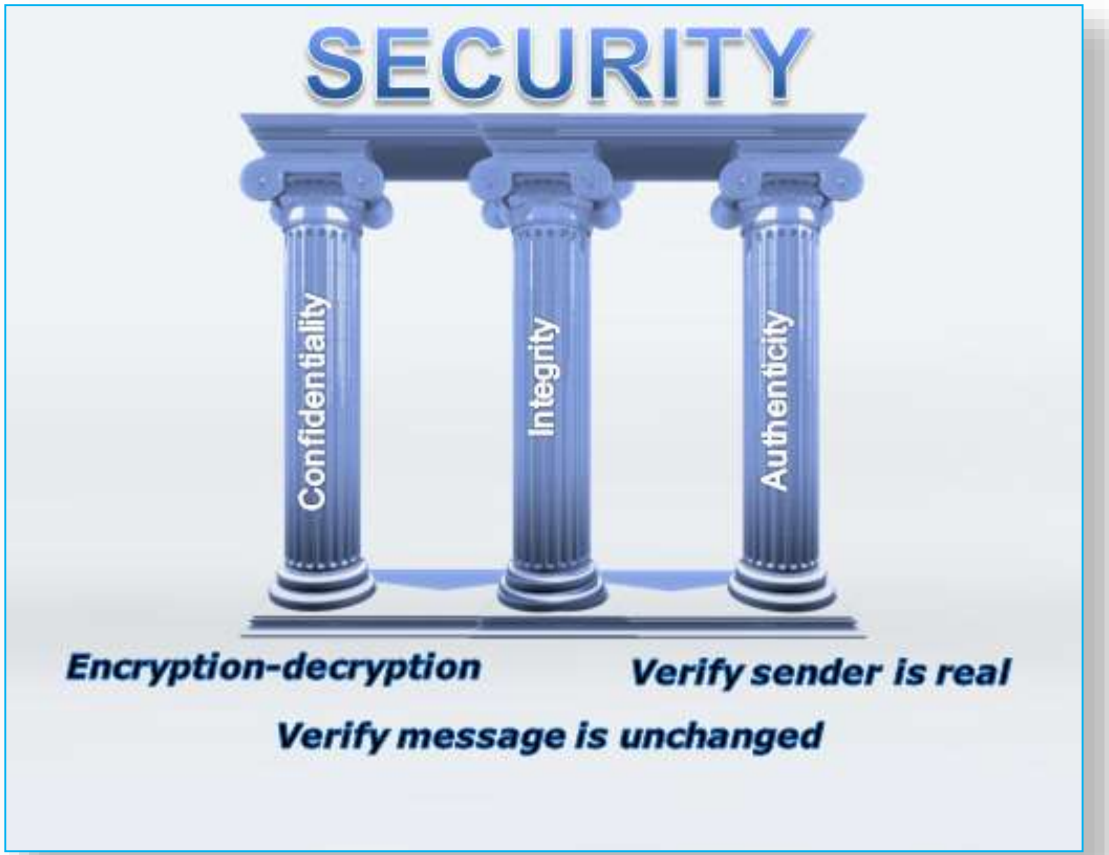
Benefits

- **Encryption algorithms are useful for coding and decoding messages**
- **Hash algorithms cannot be reversed (i.e. are one-way so the original value cannot be derived)**
- **Hashes are useful for checking the identity of the sender and the integrity of the data**

28. *Crypto Basics 4:* Real Short Glossary

- **Encryption** is encoding and decoding of a message for confidentiality. It is always reversible. Encryption does not authenticate.
- **Authentication** is used to check the identity and of the sender and the integrity of the message. The message is not necessarily encrypted.
- **Symmetric authentication** uses an identical key on the host and client sides, and both of those keys must be protected (this is very important). Symmetric authentication tends to be relatively fast.
- **Asymmetric authentication** uses a public / private key pair. The private key must be securely stored on the client. Secure key storage on the host is not needed. The keys are mathematically related, but the private key cannot be obtained from simply having the public key.   More computation is required with asymmetric authentication than symmetric.
- **Key Storage** is one of the most important determinants of the strength of security in a system. Hardware key storage is much stronger than software-based solutions because hardware storage methods are much more difficult to attack. (CryptoAuthentication ICs are hardware key storage devices.)
- **Hash functions** are a group of characters that is mathematically mapped to a value of a certain length and is representative of the original string of characters, but is normally smaller than the original. Hash functions are commonly used in authentication and encryption operations. They are one way functions which means that the original value cannot be recreated from the hashed value (i.e. digest).
- **Challenge-Response** is an operation where a challenge (usually a random number) is sent to a client to elicit a response in order to test the authenticity of a client. The response is often a hash value of that number another number such as a cryptographic key. The response is then compared to a parallel calculation done on the originating device to see if they match..
- **MAC (Message Authentication Code)** uses a secret key and cipher algorithm (e.g. SHA-256) to produce a value (called the MAC) which can be used to ensure the data has not been modified. The response of a challenge-response is a MAC.
- **Sign/Verify** is an authentication operation that creates a hash digest of data, which is then encrypted using a private key to make a signature. To verify, the receiver hashes the received data and then decrypts the received signature of that data with the sender's public key. If the signature received from the sender matches the hash that the receiver generated with the public key, then the signature is considered valid.
- **ECDSA** (Elliptic Curve Digital Signature Algorithm) an elegant and standardized method to provide asymmetric authentication using a sign-verify process using Elliptic Curve (ECC) algorithms to make a signature. ECDSA has two phases which are 1) to verify the public key, and 2) verify the private key of the client (i.e. accessory) device. ECDSA capability is built into the Atmel ATECC108A device. X.509 is an ITU standard that specifies what goes into the certificate.
- **ECDH Key Agreement** (Elliptic Curve Diffie-Hellman) an elegant and standardized method to provide asymmetric authentication using a sign-verify process using an anonymous key agreement protocol that allows two parties, each having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key which can then be used to encrypt subsequent communications using a symmetric key cipher. It is a variant of the Diffie–Hellman protocol using elliptic curve cryptography.
- **Root certificate** is a public key certificate that identifies the Root Certificate Authority (CA). The most common commercial variety is based on the ITU-T X.509 standard, which normally includes a digital signature from a certificate authority. .Digital certificates are verified using a chain of trust. The trust anchor for the digital certificate is the Root Certificate Authority (CA).
- **Public-key cryptography**, is a class of algorithms which requires two separate keys, one of which is secret (private) and one public. This key pair is mathematically linked by algorithms that are computationally infeasible to determine the private key from the public key. The public key is used to encrypt plaintext or verify a digital signature. The private key is used to decrypt ciphertext or to create a digital signature. The term "asymmetric" stems from the use of different keys to perform these opposite functions, each the inverse of the other – as contrasted with conventional ("symmetric") cryptography which relies on the same key to perform both. Public-key algorithms are based on mathematical problems which currently admit no efficient solution that are inherent in certain integer factorization, discrete logarithm, and elliptic curve relationships.

29. *Crypto Basics 5:* "CIA" The Three Pillars of Security



To provide the full set of security you will need all three of the foundational pillars of security: Confidentiality, Integrity (of data), and Authentication. These can be remembered as "CIA":

Confidentiality

This is ensuring that no one can read the message except the intended receiver. This is typically accomplished with encryption and decryption which hides the message from all parties but the sender and receiver.

Integrity

This is also called data integrity and is assuring that the received message was not altered. This is done using cryptographic functions. For symmetric this is typically done by hashing the data with a secret key and sending the resulting MAC with the data to the other side which does the same functions to create the MAC and compare. Sign-verify is the way that asymmetric mechanisms ensure integrity.

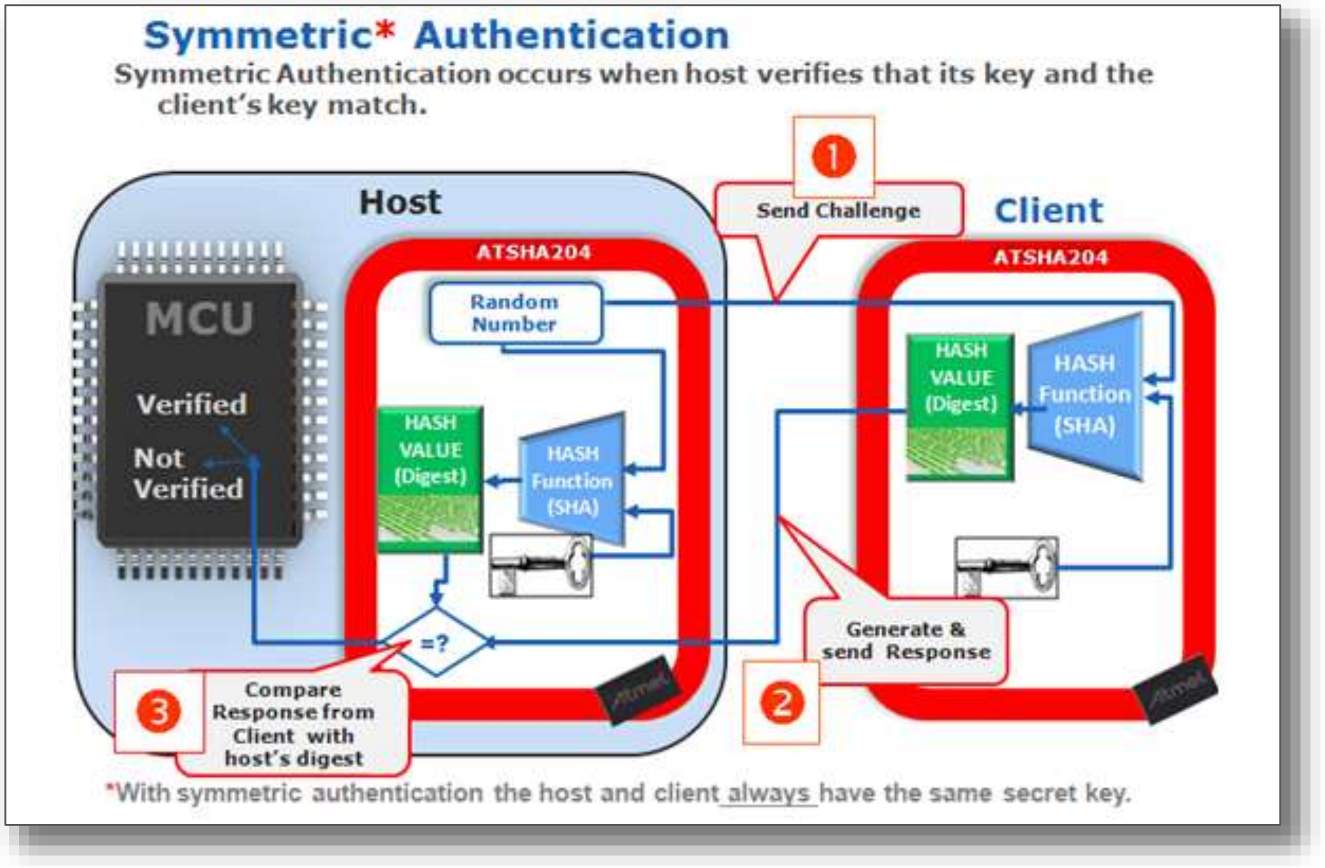
Authenticity

This is verification that the sender of a message is who they say they are (i.e. are real). In symmetric authentication mechanisms this is usually done with a challenge (often a random number) that is sent to the other side that is hashed with a secret key to create a MAC response which then gets sent back to run the same calculations and then compare the response MACs from both sides.

Sometimes people add **non-repudiation** to the list of pillars which is preventing the sender from later denying that they sent the message in the first place.

30. *Symmetric Authentication* with secrets stored on the host using Challenge-Response.

With Symmetric Authentication using secrets stored on the host using Challenge-Response, like you might expect from the name, the host sends a challenge and the client makes a response using cryptographic "magic" (i.e. math) and then the host runs the same process in order to make a comparison to see if it gets the same answer. It the answers on each side match then the client is real.

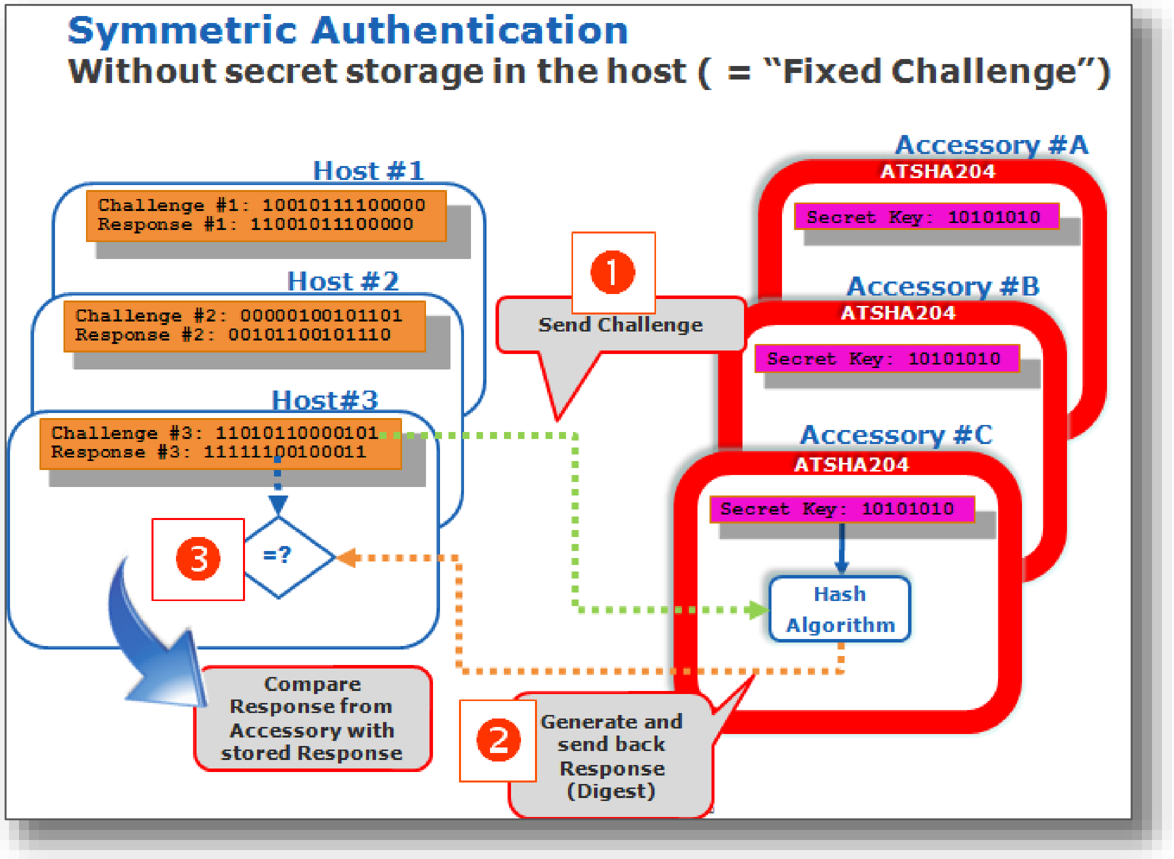

Symmetric* Authentication
Symmetric Authentication occurs when host verifies that its key and the client's key match.
1
Send Challenge
Host
Client
ATSHA204
ATSHA204
MCU
Verified
Not Verified
Random Number
HASH VALUE (Digest)
HASH Function (SHA)
HASH VALUE (Digest)
HASH Function (SHA)
=?
2
Generate & send Response
3
Compare Response from Client with host's digest
*With symmetric authentication the host and client always have the same secret key.

The process starts when the host sends a random number, which is generated by the ATSHA204's random number generator to the client. It does this at the time that it wants to verify if the client is real such as when an ink cartridge is inserted into a printer. This step is called the "Challenge" and is shown as step one. The client receives the random number challenge and runs it thru a hash algorithm (SHA256) using the secret key stored there. The result of the hashing function is called the "Response" and also called "Message Authentication Code" (or MAC). The response is then sent to the host, and together these actions comprise step two. Moving to step three, the host internally runs the same challenge number (i.e. the random number) it sent to the client thru a hash algorithm using the secret key stored on the host side. Then the host compares the hash value calculated on the host side with the response hash value sent from Client. If the two hash values match then the Client is verified. And that is how to do symmetric verification using the ATSHA204 on both sides.

Benefits

- **Symmetric authentication is fast**
- **Crypto devices on both host and client sides ensures very secure secret storage**

31. *Symmetric Authentication* without secret storage on the host using a Fixed Challenge


Symmetric Authentication
Without secret storage in the host (= "Fixed Challenge")
Host #1
Challenge #1: 10010111100000
Response #1: 11001011100000
Host #2
Challenge #2: 00000100101101
Response #2: 00101100101110
Host#3
Challenge #3: 11010110000101
Response #3: 11111100100011
=?
3
Compare Response from Accessory with stored Response
1
Send Challenge
Accessory #A
ATSHA204
Secret Key: 10101010
Accessory #B
ATSHA204
Secret Key: 10101010
Accessory #C
ATSHA204
Secret Key: 10101010
Hash Algorithm
2
Generate and send back Response (Digest)

Symmetric authentication can also be accomplished without having a crypto device on the host side.

Such an arrangement is called is fixed challenge. That means that the host does not use a random number, but instead use a fixed number pair, or series of pairs, of specific challenge and response numbers that are programmed into the host's memory. To calculate the response value each challenge value is run through a hash algorithm using the same secret key (or keys) in the target client (or clients).   The challenge and corresponding response values are loaded and stored on the host.

When the host sends its preloaded challenge to the client to check if the client is real or not (which is step 1), the client will run the hash algorithm on that challenge number and generate a response.

It then sends that response back (which is step 2).

The host will compare the response from the clients with the pre-loaded response value stored in its memory (which is step 3).

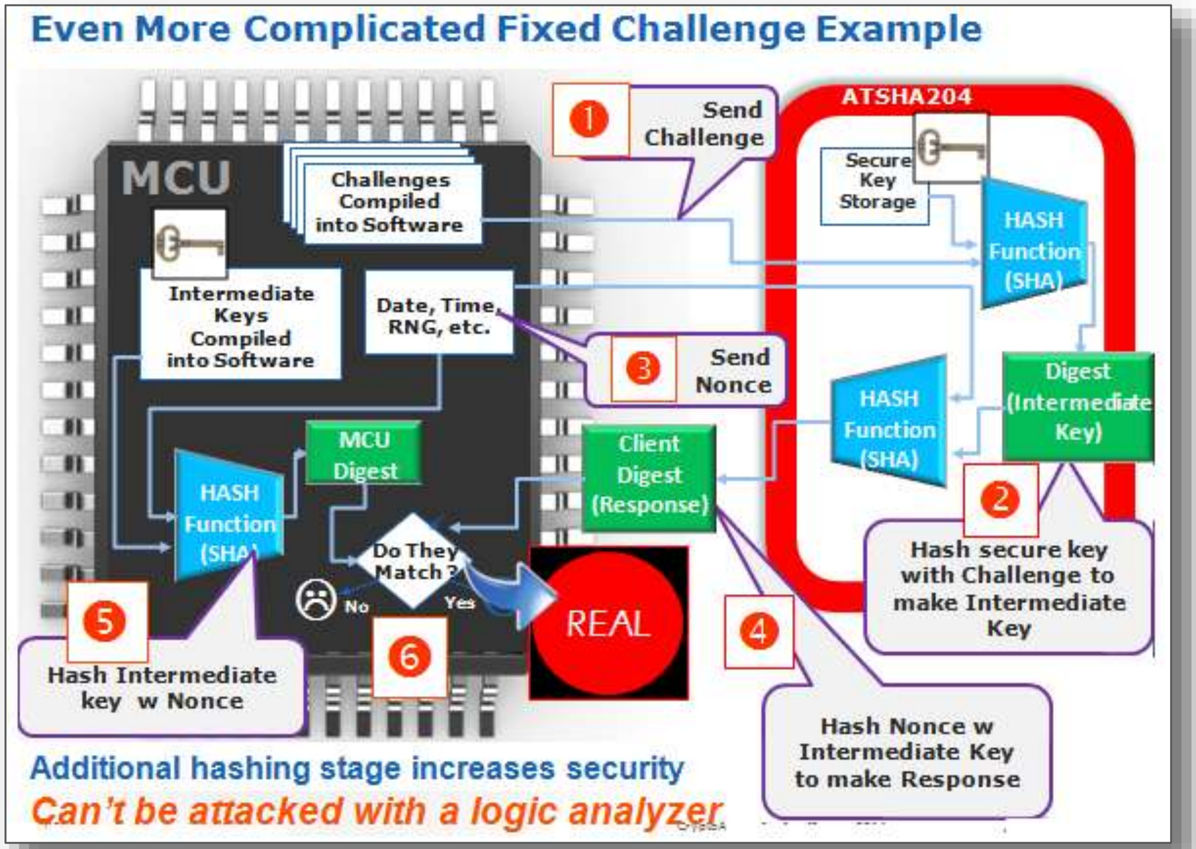
If the client is real then the response from the client (which is the hash value based on the secret key and the challenge) will be the same as the response value that was preloaded in the host.   This approach can be used for firmware protection, designs with no secrets in the host (as noted), and can be implemented with very low cost MCUs.

Benefits

- **Symmetric authentication is fast**
- **No secrets in the host**
- **Can use low cost MCU of host because less computation is needed for a fixed challenge**

32. *Symmetric Authentication* without secret storage on the host using Fixed Challenge and intermediate key.

Fixed challenges are great if low cost micros are desired, however, some security is sacrificed since a logic analyzer can be employed to break the security. Fortunately, there is an easy way to add more security to fixed challenge scenario, and that is by using an additional hashing stage with an intermediate key. The process starts with the challenge that is compiled into the MCU's software. That challenge is hashed with the secret key stored on the client creating the intermediate key. The intermediate key is then hashed with a unique number such as a random number or the date-time-etc. from issued by the MCU. That unique number is used just once and therefore it is called the "NONCE", which means number used one. The hash of the intermediate key and the NONCE becomes the client's response. That response is compared to a digest on the MCU (.i.e. "MCU Digest") which is created on the MCU using the NONCE which gets hashed with the intermediate key compiled into the MCU's software. The MCU Digest will match the Client Digest (i.e. Response) if the client is real.


Even More Complicated Fixed Challenge Example
MCU
Challenges Compiled into Software
Intermediate Keys Compiled into Software
Date, Time, RNG, etc.
MCU Digest
HASH Function (SHA)
Do They Match?
No
Yes
REAL
1 Send Challenge
ATSHA204
Secure Key Storage
HASH Function (SHA)
Digest (Intermediate Key)
HASH Function (SHA)
3 Send Nonce
Client Digest (Response)
2 Hash secure key with Challenge to make Intermediate Key
4 Hash Nonce w Intermediate Key to make Response
5 Hash Intermediate key w Nonce
6
Additional hashing stage increases security
Can't be attacked with a logic analyzer

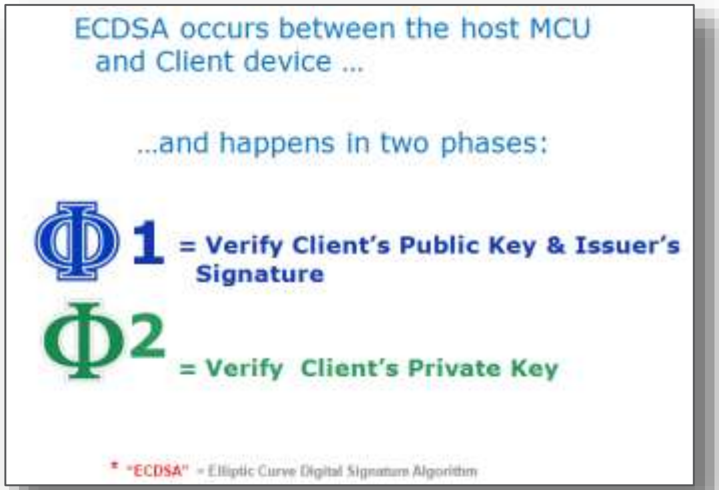
Note that the response will change each time since the NONCE is different each time, and that is what makes this more secure. Trying to analyze the responses with a logic analyzer will be fruitless.

Benefits

- **Cannot be attacked with a logic analyzer**
- **Increased security**

33. *Asymmetric Authentication* using Digital Signatures (ECDSA)

In the real world the Asymmetric Authentication process typically begins when a client device is inserted into a host system or the host system wants to know what exactly is connected to it. Examples are a printer ink cartridge being inserted into a printer, a thermostat control block wanting to talk to a remote temperature sensor, a cell phone connecting to a wall charger, and many others.



ECDSA occurs between the host MCU and Client device ...

...and happens in two phases:

Φ1 = Verify Client's Public Key & Issuer's Signature

Φ2 = Verify Client's Private Key

* "ECDSA" = Elliptic Curve Digital Signature Algorithm

The best way to understand how ECDSA performs authentication and how it ties that back to the root of trust is to break it down into its individual steps. Admittedly, there are many steps but each one does a very specific and simple thing, so it is easy to follow. To simplify the process, it is useful to group the steps into sequential phases that perform distinct objectives in the process. Fortunately, there is a clear break between two major objectives, so we can break it into a phase one and phase two**.**

Phase one's goal is to use ECDSA calculation algorithms to verify the public key on the client. Phase one has a second goal which is implied, and that is to verify the entire certificate chain back to the root of trust (i.e. the certificate authority or "issuer"). One way to look at it is that the host MCU must identify and confirm the entire history of who signed what. (In this example we will look at a two level signing history going from the issuer to the signing module to the client device.)

To accomplish the verification of the public key in phase one it is necessary to verify all the signatures in the certificate chain. In this example the signatures are stored in the client in two distinct digital certificates. One is the client's certificate and the other is the signer's certificate. The certificates are the mechanism that allows the chain to reach back to the root of trust (i.e. the issuer).

Phase two's goal is to use ECDSA calculations to verify that the private key on the client is related to the previously verified public key. Verifying that the client has a valid public-private key pair is the soul of symmetric authentication. If the ECDSA verify calculation operations of both phases pass then the client is verified as real. And that is the whole purpose.

Benefits

- **Increased security because asymmetric authentication does not need secure key storage on the host (only the client)**
- **No need to update the host with secrets in the field. (Can update the public key at any time.)**
- **ATECC108 has ECDSA built in, making it easy to implement.**

34. *Asymmetric Authentication*: Making the ECDSA certificates

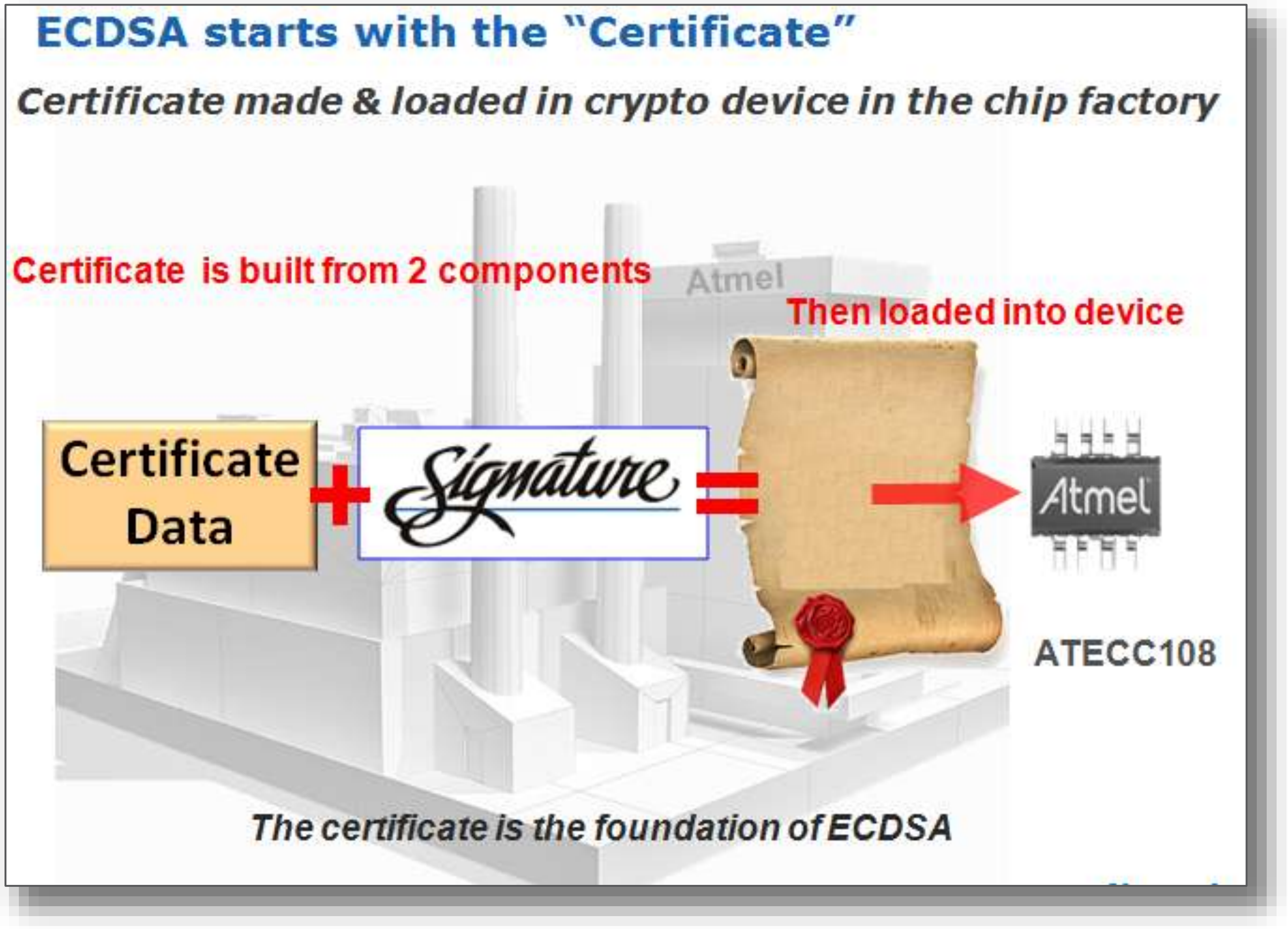
To understand the ECDSA process it is very helpful to first look at how the certificates are built and loaded into the crypto device. This happens in the factory.

A certificate is made from two components:

1) The certificate data
2) The signature.

The client certificate creation process begins in the factory on the machine that tests the crypto device: in other words, the tester. Attached to the tester is equipment called the "signing module" that contains its own secret private key. That private key is securely stored and never shared. The signing module is used to create the signature, just as its name implies. Once the certificate is made it is loaded into the device.

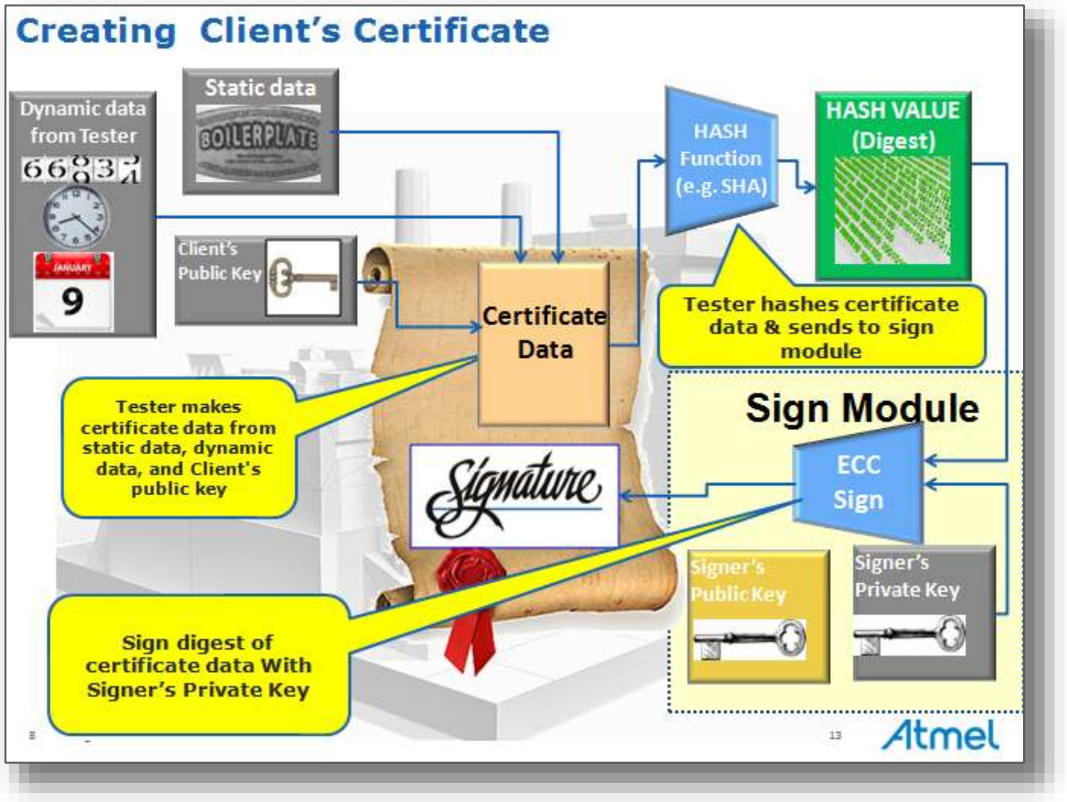
Now let's look at those steps:


ECDSA starts with the "Certificate"
Certificate made & loaded in crypto device in the chip factory
Certificate is built from 2 components
Atmel
Then loaded into device
Certificate Data
Signature
Atmel
ATECC108
The certificate is the foundation of ECDSA

35. *Asymmetric Authentication*: Creating the Client's Certificate

First, let's look at the creation of the certificate data.

The certificate data is made up of three items: 1) static data, 2) dynamic data, and 3) the client's public key. You can look at the static data as a type of boilerplate that contains basic information such as the company's name, address, and other information that never changes. In contrast, dynamic data from the tester are data that generally change with each part or group of parts being tested. Dynamic data include things like the serial number, date, time, expiration date, and so on. The client's public key is the third item that goes into the certificate data. Recall that the client's public key is paired to the private key of the client device. The private key is securely stored in the ATECC108 and never shared (which makes it private). Now the process moves to making the signature.

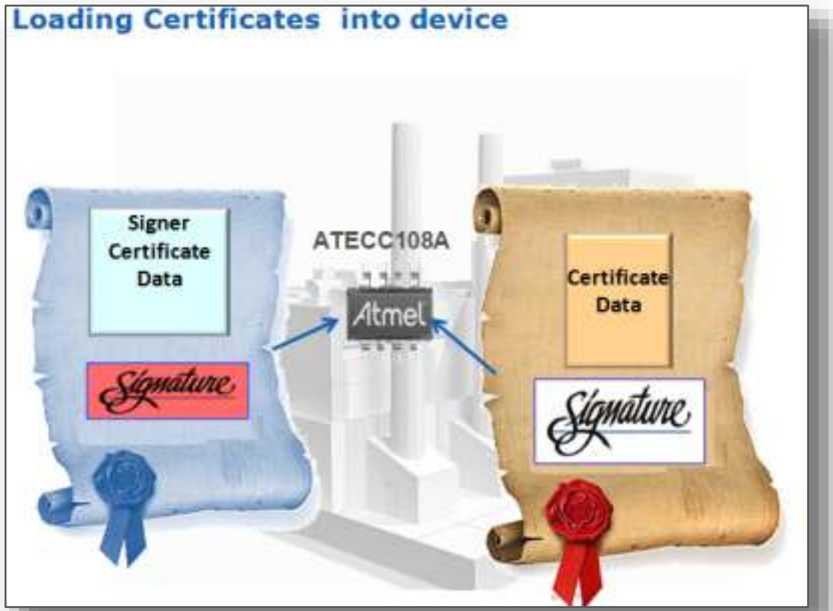

Creating Client's Certificate
Dynamic data from Tester
6683
JANUARY
9
Static data
BOILERPLATE
Client's Public Key
Certificate Data
HASH Function (e.g. SHA)
HASH VALUE (Digest)
Tester hashes certificate data & sends to sign module
Tester makes certificate data from static data, dynamic data, and Client's public key
Signature
Sign Module
ECC Sign
Signer's Public Key
Signer's Private Key
Sign digest of certificate data With Signer's Private Key
Atmel

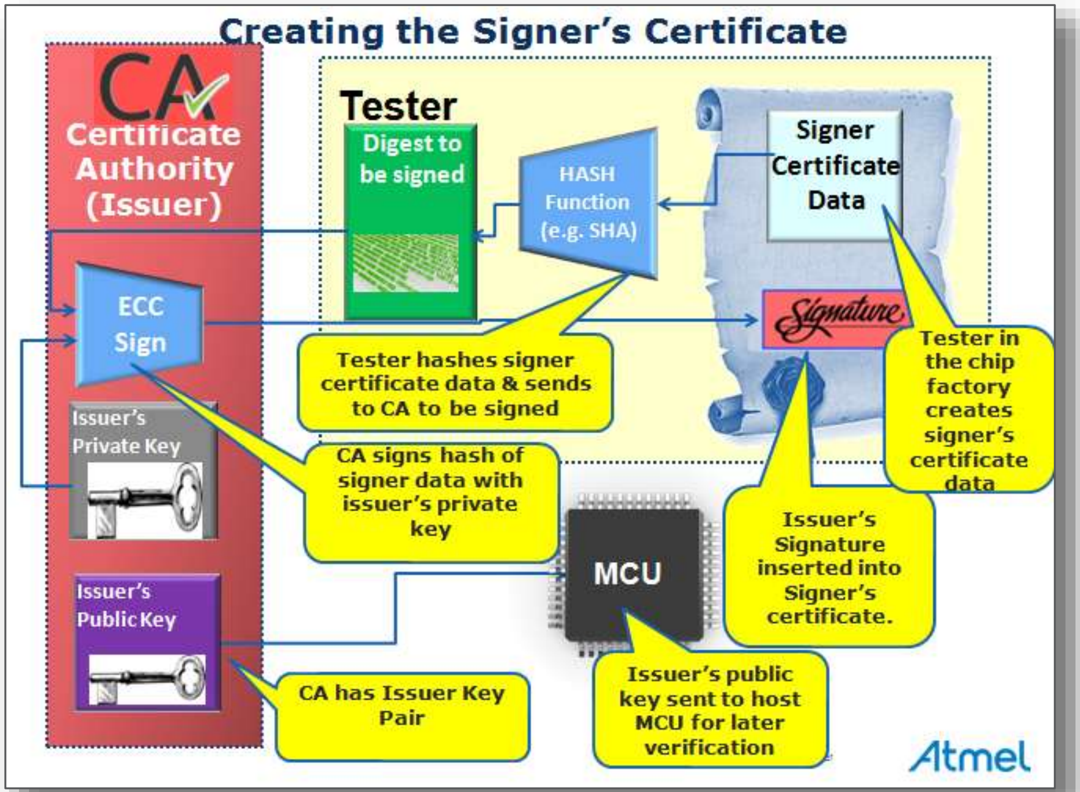
Recall that the certificate data comprises just half of a certificate. The other half is the signature. What is a little tricky to understand at first is that the certificate data have two purposes when it comes to building the certificate: (1) to become part of the certificate, and (2) to get hashed and then run through a signing algorithm to produce the signature. Both the certificate data and the signature made from that certificate data make up the complete certificate, and that is a major point to remember. You can see the two purposes of the certificate data clearly in the diagram, which shows how the certificate data are assembled and stored in the certificate and then also digested and input to the signing process on the signing module.

As for the details, the signature process begins with a copy of the certificate data being put through a hash algorithm to create a number called a hash value (or digest). ECDSA P-256 specifies a 32 byte digest length and SHA256 as the hashing algorithm. Once created, the digest is ready to be signed by the sign module in the factory.
The sign module is a piece of equipment that securely stores the signer's private key. Being securely stored means that no one can get access to that key. The sign module uses the ECC sign algorithm to sign the digest of the certificate data with the signer's private key. The result of that process becomes the "signature" of the certificate data that went into the singing module and signed with the private key of the module. The signature then joins the original (i.e. unhashed) certificate data to complete the certificate.

36. *Asymmetric Authentication*: Creating the Signer's Certificate

Another certificate called the signer's certificate is used to create the link to the certificate authority (issuer). The signer's certificate is made by the issuer creating a signature over the signer's certificate data that it receives from the tester. It signs the data using its (i.e. the issuer's) private key. Both the signer's and client's certificates are stored in the ATECC108A device. The signer's certificate is made by the tester's assembling the signer's certificate data and placing it into the certificate, and then hashing signer certificate data and sending that digest to the certificate authority (issuer) to be signed with the issuer's private key. Once created, the signature is sent back to the tester to be inserted into the signer's certificate alongside the signer's certificate data. That completes the singer's certificate. The public key of the issuer that corresponds to the issuer's private key gets sent to the MCU to use later during the actual authentication process. Both certificates are now finished and can be securely installed into the crypto device by the tester.




Creating the Signer's Certificate
CA
Certificate Authority (Issuer)
ECC Sign
Issuer's Private Key
Issuer's Public Key
Tester
Digest to be signed
HASH Function (e.g. SHA)
Signer Certificate Data
Signature
Tester hashes signer certificate data & sends to CA to be signed
CA signs hash of signer data with issuer's private key
Tester in the chip factory creates signer's certificate data
Issuer's Signature inserted into Signer's certificate.
MCU
Issuer's public key sent to host MCU for later verification
CA has Issuer Key Pair
Atmel

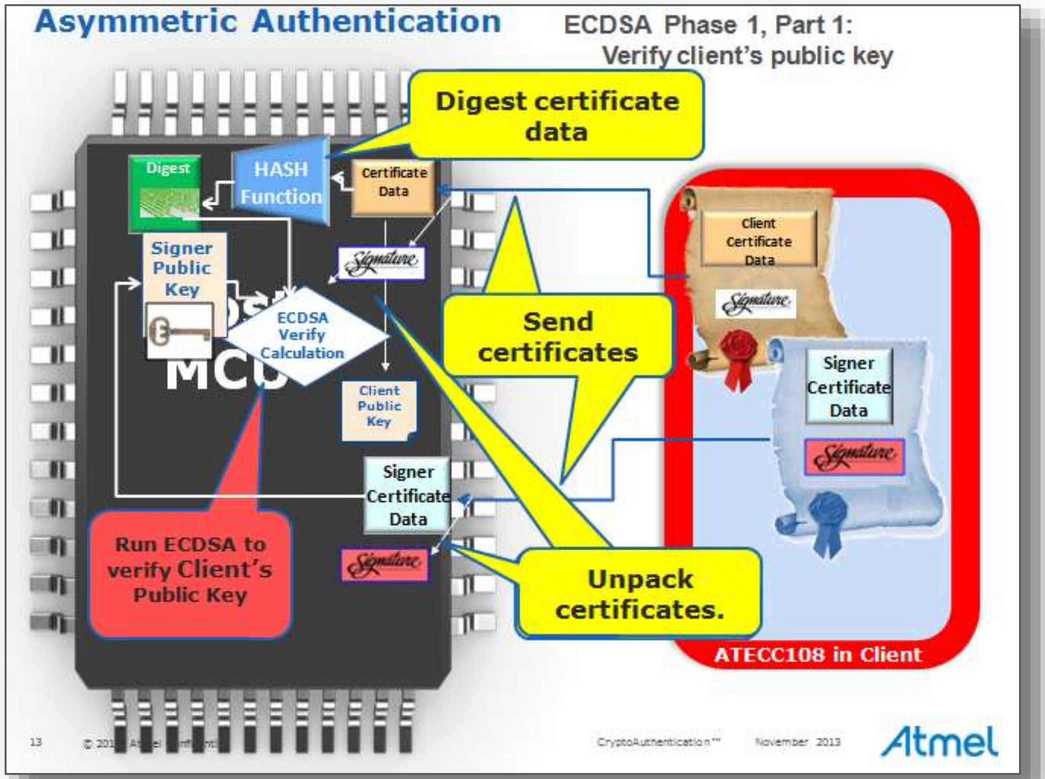
37. *Asymmetric Authentication* ECDSA is a two phased process (Phase 1, Part 1: Verify Client's Public Key)

ECDSA is a two phased process with phase one being to verify both the client's and signer's public keys. Phase one links (i.e. chains) the client's and signer's signatures back to the issuer's signature, which establishes the root of trust (i.e. ties to the anchor).

Phase two's purpose is to verify the client's private key. When each of the keys are verified it is proven that there is a valid client public and private key pair, and that they tie back to the root of trust. In short, that means that the client is mathematically verified as being real (i.e. authenticated).

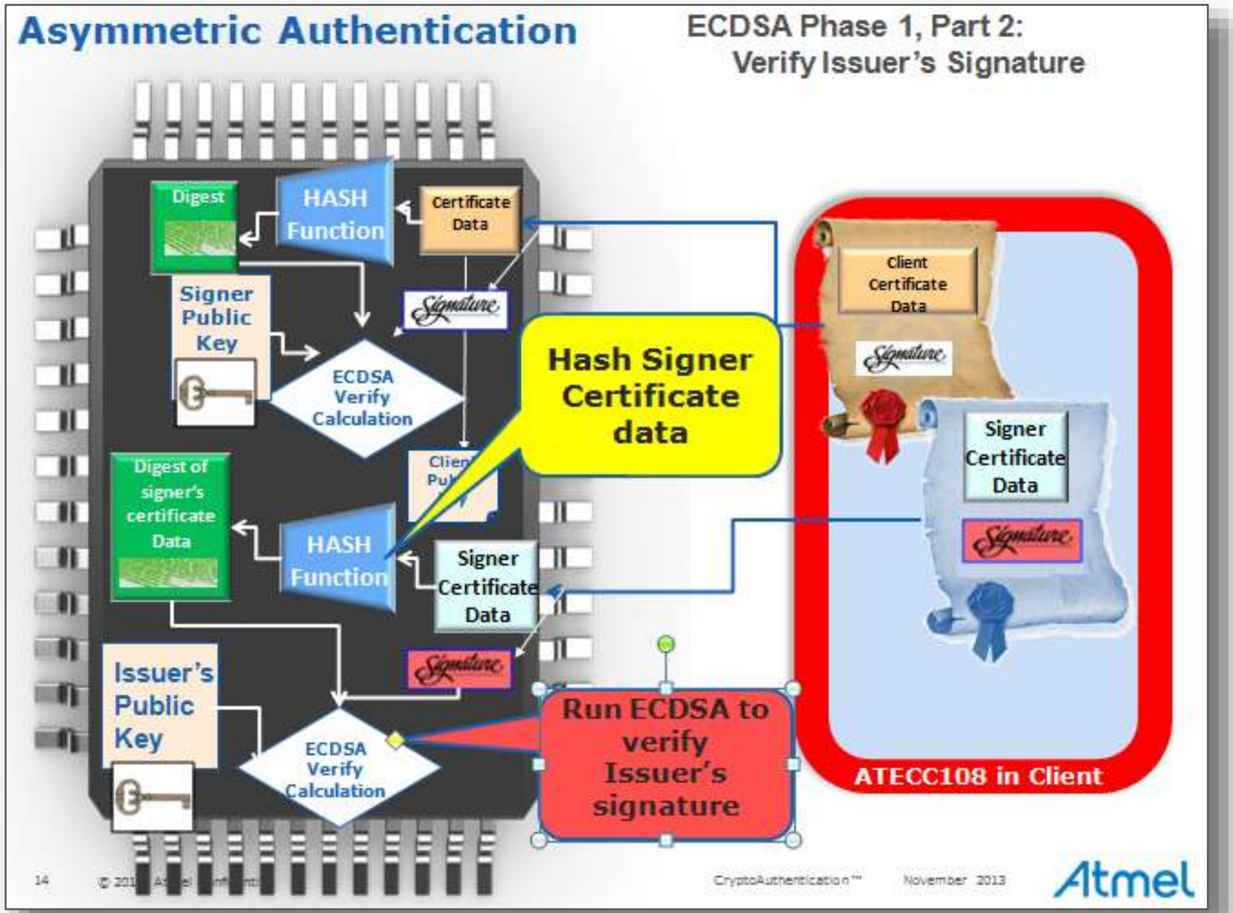
Phase 1 starts with the host requesting information to be sent over by the client (accessory). That information comes over to the host in the client's certificate and in the singer's certificate. When the host receives the certificates, it extracts the client's certificate data (client's static data, client's dynamic data, and client's public key) and the signature made by the signing module in the chip factory) from the client's certificate. It also extracts the signer's certificate data (including the signer's public key) and the issuer's signature that was made by the certificate authority (the issuer). The host runs a hashing process on the both sets of certificate data that it just received, creating two 32-byte message digests (P-256 curve): 1) the client's digest, and 2) the signer's digest.

The extracted signer's public key that came over in the signer's certificate is input to the ECDSA verify calculation together with the client's digest (number 1 above) and the client's signature extracted from the client's certificate. The purpose of this ECDSA calculation is to verify the client's public key.



38. *Asymmetric Authentication*: ECDSA is a two phased process (Phase 1, Part 2: Verify Issuer's Signature)

Before the prior phase one ECDSA calculation can be accepted as complete verification of the client's public key, another ECDSA calculation (phase 1, part 2) must be run to verify that the root of trust exists. The root of trust is established by linking of the signer's signature to the issuer's signature. This is done by verifying the issuer's signature that came over in the signer's certificate.



To do so, the signature in the signer's certificate (which is the issuer's signature) is input to the second ECDSA calculation along with two other inputs:

1) The digest made by hashing the signer's certificate data, and

2) The issuer's public key that came over to the MCU at some earlier point (and was stored in memory of the MCU).

If both ECDSA calculations pass then the client's public key is considered to be verified all the way back to the root of trust (i.e. the issuer).

39. *Asymmetric Authentication:* ECDSA is a two phased process (Phase 2: Verify Private Key)

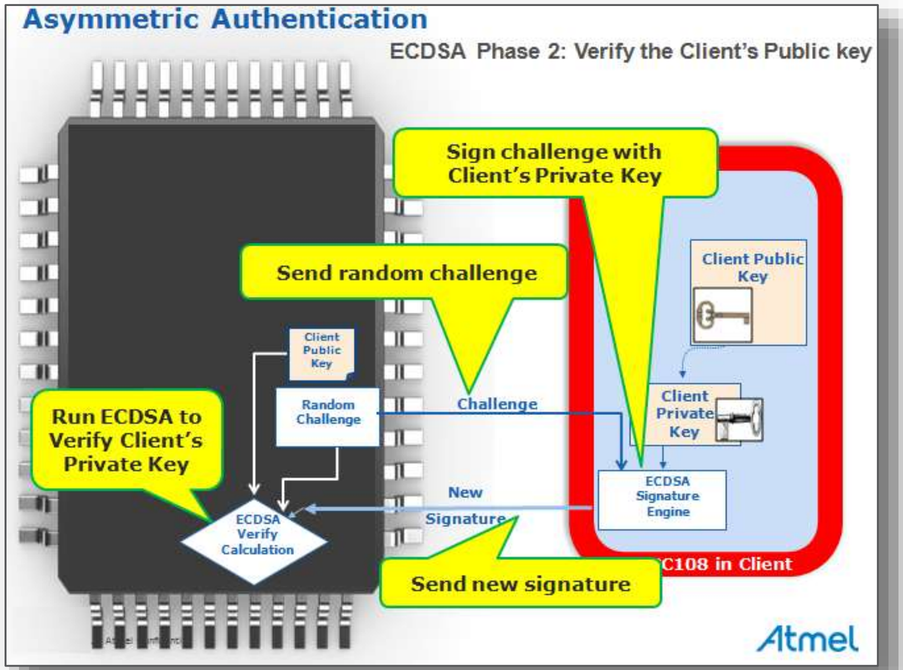
When both phase 1 ECDSA calculations pass, then phase 2 starts with the goal of verifying the client's private key. Recall that the whole point of this two-phased process is to verify mathematically that the client's private key and public key are indeed a valid key pair and tie back to the root of trust.

Phase 2

This phase begins with the host generating a random number challenge and sending it to the client. The client device uses the ECDSA signature engine in the ATECC108A to create a new signature using this random number and the client's (secret) private key securely stored there. That new signature is then sent to back the host, which uses it along with the same random number used to make the signature and the client's public key (that was verified in phase one) as the inputs to the Phase 2 ECDSA verify calculation. If that ECDSA calculation succeeds, then the host has then proven that the accessory (client) is real (i.e. that the client contains a valid private-public key pair). As you can see, the ATECC108A does all the heavy mathematical lifting.

In addition, Atmel provides the tools that Atmel provides make it easy to program the microcontroller to do its part without having to securely store a secret. That is the whole reason for asymmetric authentication: the secret only has to be secured on one side.

The engineering and mathematics behind authentication using sophisticated algorithms may not be easy, but that does not matter to the user because Atmel makes it easy to implement cryptography without having to be a cryptography expert.

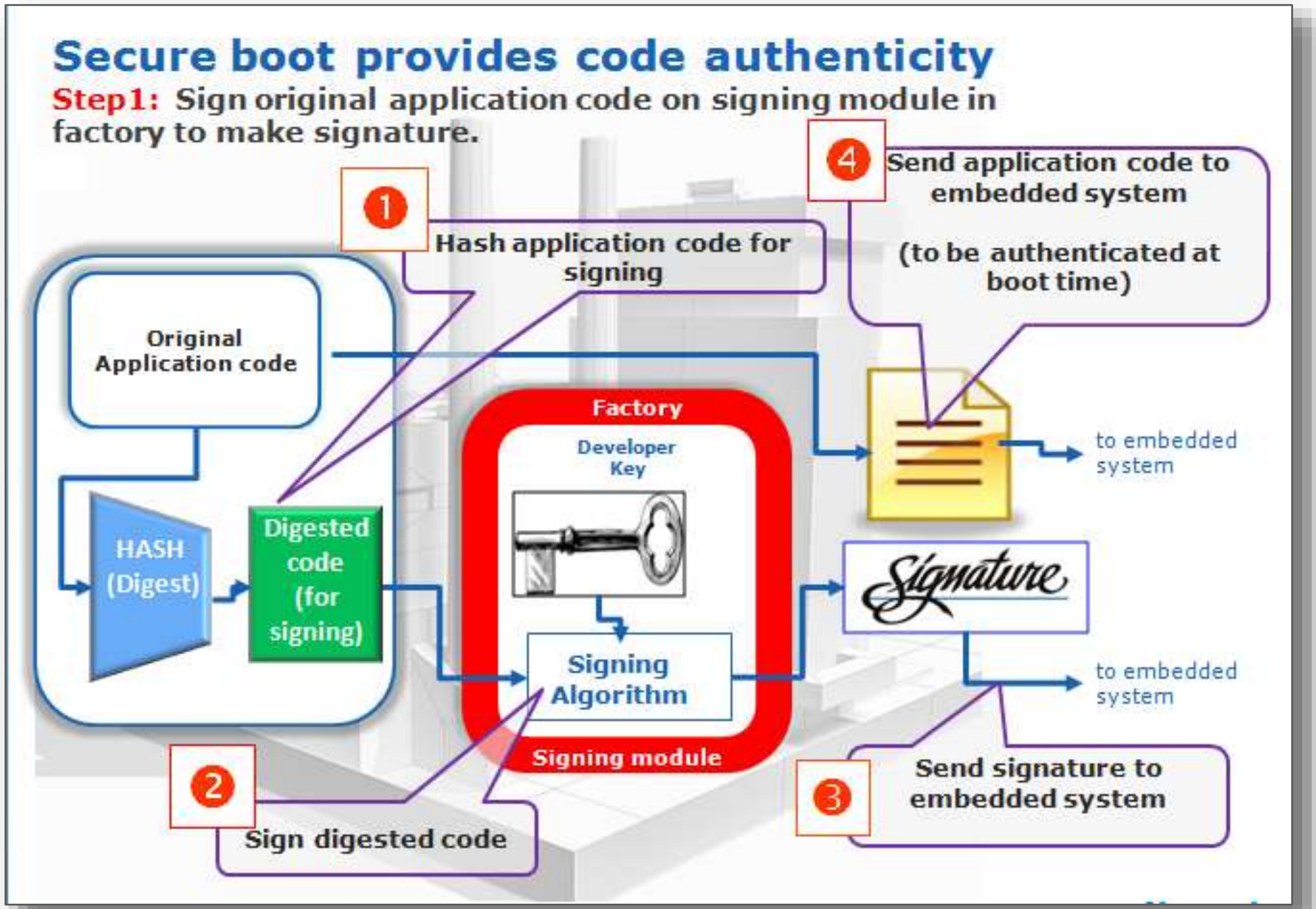


Benefits

- **ECDSA is a proven and secure authentication process**
- **Uses the advantages of Elliptic Curve Cryptography (high security , short key, less computation)**
- **No need for secure key storage in the host**

40. *Firmware Protection:* Secure Boot (Step 1. Making the signature in the factory)

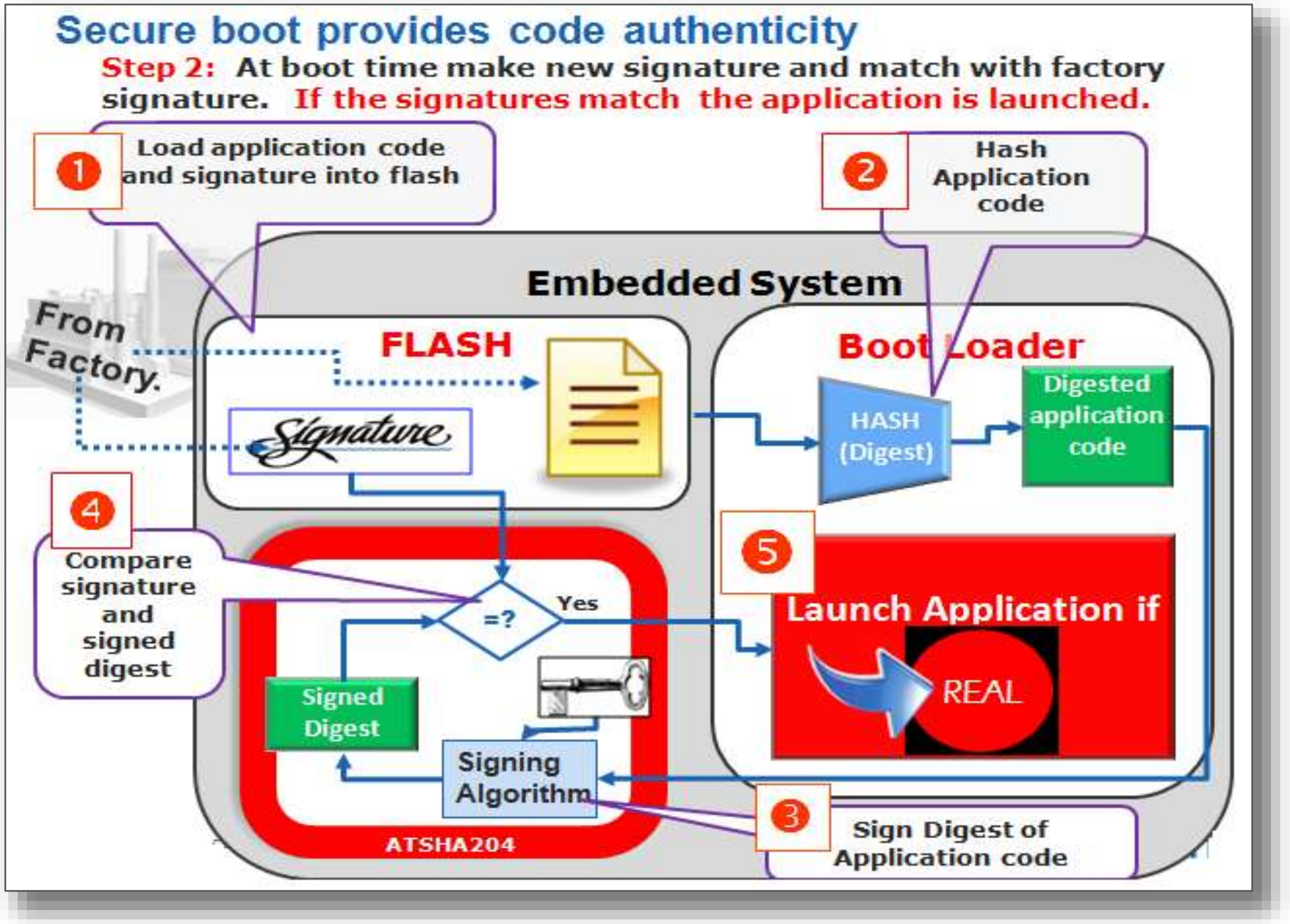
Authentication is useful for many applications, such as securing data storage by encrypting stored data with unique key for each file. Another application is authenticating stored data such as the manufacturer's serial number, manufacturing data logs, configuration parameters, and calibration data. CryptoAuthentication devices are ideal for implementing secure session key exchange because the private (secret) key never leaves the security device allowing the host MCU to encrypt the data stream with ephemeral session key. CryptoAuthentication is one of the most secure ways for handling passwords because comparisons are made internally, so attackers cannot find the expected value, also passwords can be mapped to a high-entropy (i.e. highly randomized) key(s).


Secure boot provides code authenticity
Step1: Sign original application code on signing module in factory to make signature.
1 Hash application code for signing
2 Sign digested code
3 Send signature to embedded system
4 Send application code to embedded system (to be authenticated at boot time)
Original Application code
HASH (Digest)
Digested code (for signing)
Factory
Developer Key
Signing Algorithm
Signing module
Signature
to embedded system
to embedded system

Step 1 is the signing of the application code in the factory. The code and the signature is sent to the embedded system in the field and loaded into the system memory.

41. *Firmware Protection:* Secure Boot (Step 2. Authenticating the device in the field)

Step 2 is the process of using the signature and the code itself at boot time to authenticate the code (i.e. to check if it is real). The basic idea is to use the crypto device like the ATECC108A to create a signature of the code just like was done in the factory and then compare that new signature with the one from the factory. If those two signatures match then the code is proven to be the same as the original code (i.e. not altered or fake) and is considered real and thus can be loaded.

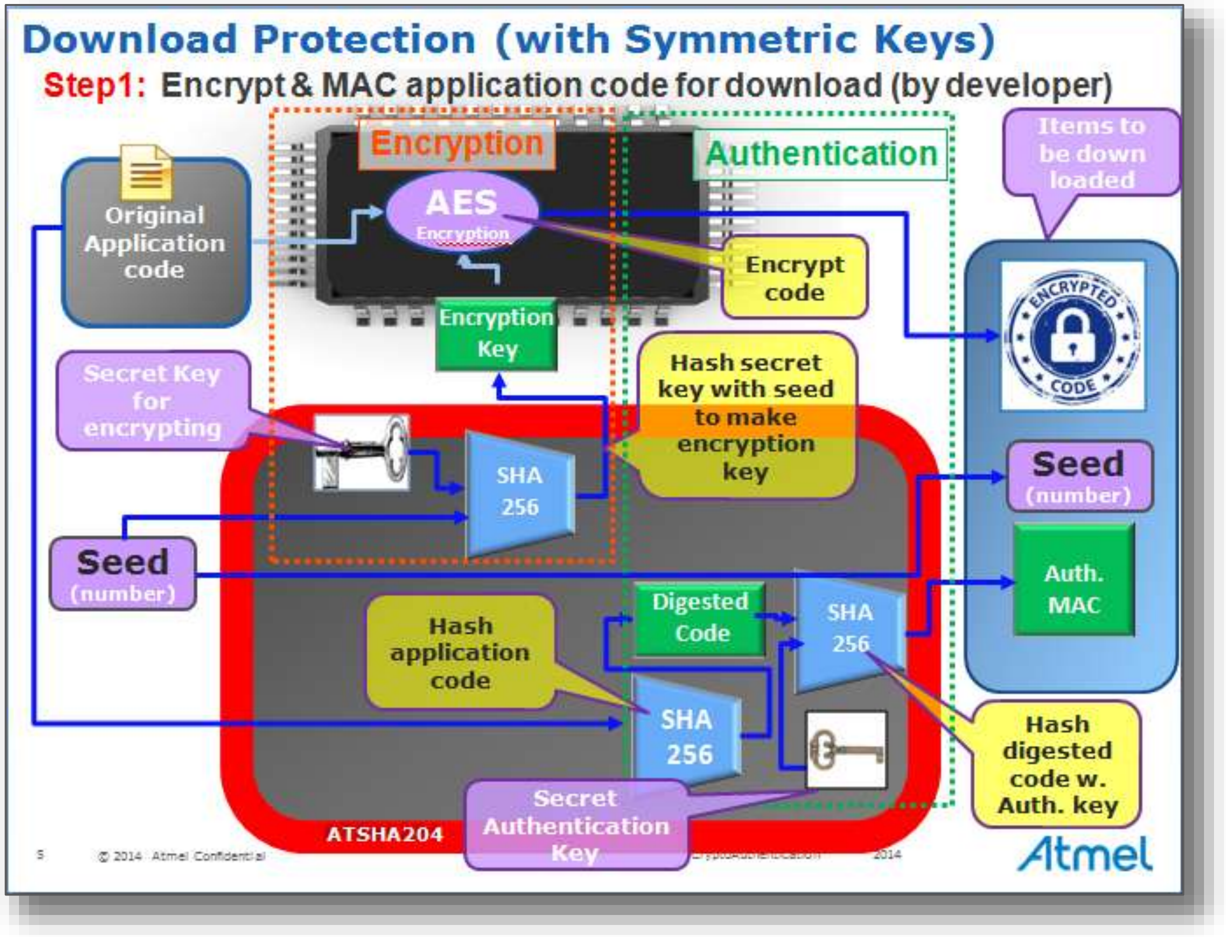

Secure boot provides code authenticity
Step 2: At boot time make new signature and match with factory signature. If the signatures match the application is launched.
1 Load application code and signature into flash
2 Hash Application code
From Factory.
Embedded System
FLASH
Signature
Boot Loader
HASH (Digest)
Digested application code
4 Compare signature and signed digest
=?
Yes
5 Launch Application if REAL
Signed Digest
Signing Algorithm
ATSHA204
3 Sign Digest of Application code

Benefits

- **Secure boot using a crypto device can protect IP and revenue streams**
- **Easy to implement very strong security**
- **Numerous applications**

42. *Protecting Downloaded Firmware* with Symmetric Keys; Step1: Encrypt and MAC application code

Software and firmware IP protection possibly provides the largest opportunity application target market. This segment includes anyone that has an embedded system or software solution. And, that is just about anyone that has a processor in their system.



Download Protection (with Symmetric Keys)
Step1: Encrypt & MAC application code for download (by developer)
Encryption
Authentication
Items to be down loaded
Original Application code
AES Encryption
Encrypt code
Encryption Key
Hash secret key with seed to make encryption key
ENCRYPTED CODE
Secret Key for encrypting
SHA 256
Seed (number)
Seed (number)
Auth. MAC
Digested Code
SHA 256
Hash application code
SHA 256
Hash digested code w. Auth. key
Secret Authentication Key
ATSHA204
© 2014 Atmel Confidential
Atmel

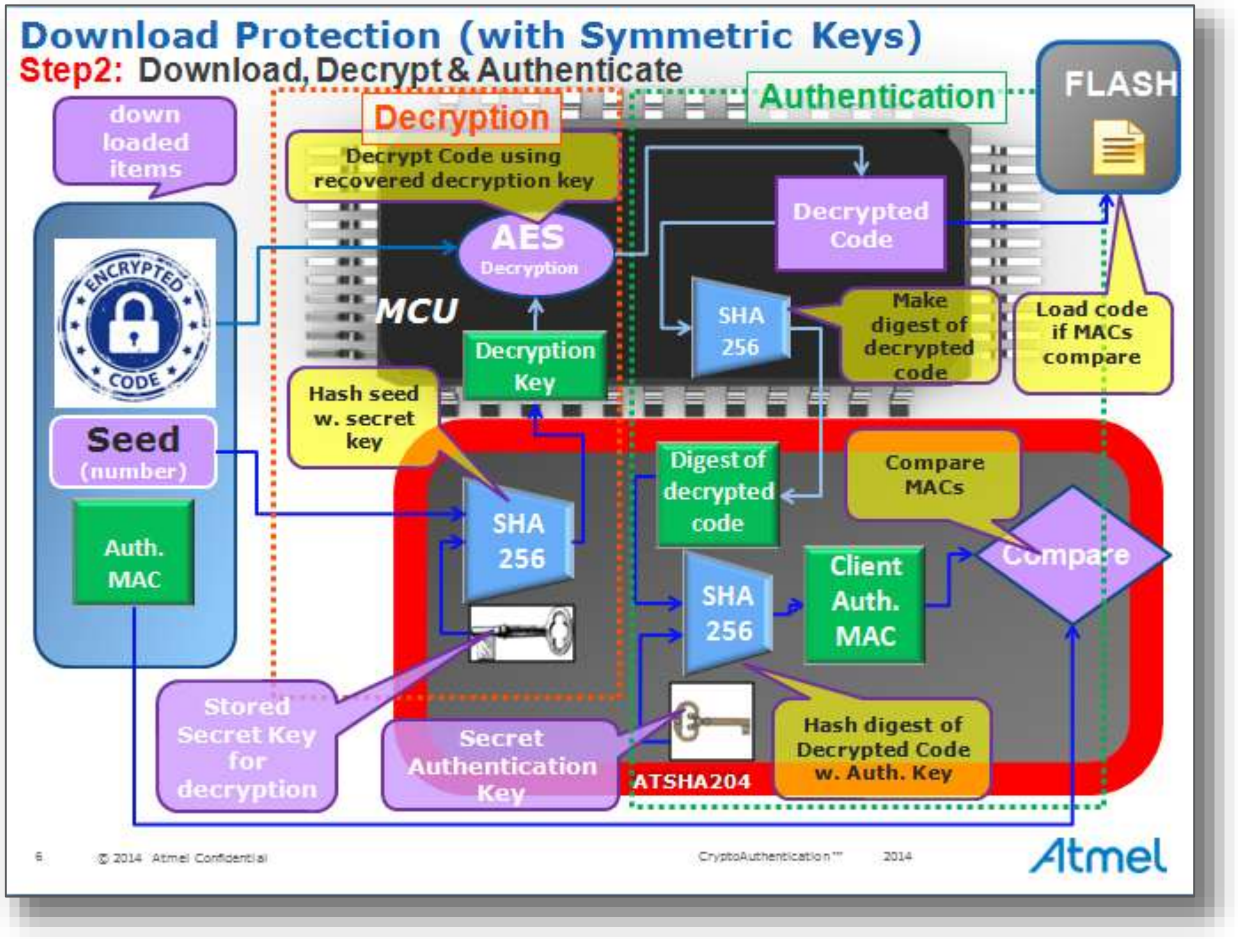
Most systems use nonvolatile memory to store programs to remotely add features and fix problems, but they are vulnerable to hacking. That is because all software based systems can be hacked. CryptoAuthentication devices protect downloaded firmware by preventing hackers from getting software images and algorithms. This ensures that only unmodified OEM-authorized firmware gets loaded in nonvolatile memory. The usage models are flexible. All downloads will be identical and the user can post download on the web without concern of theft because the firmware will be encrypted.

Two cryptographic pillars are used in this case: encryption and authentication. Encryption allows the software IP to be sent to target users without others being able to use it. It will of course be decrypted on the other side. Authentication is then used to ensure that the code once decrypted is indeed the intended code. To encrypt the original application code the developer will need to create an encryption key. This is done using an ATSHA204 that stores secret key and hashes it with a seed number. The digest or Message Authentication Code (MAC becomes the encryption key that is input to the encryption algorithm. The developer then encrypts the code which is now ready to be downloaded by the target user. The seed number will be sent with the encrypted code to allow decrypting by the target user. The other pillar (authentication) is prepared by hashing (or padding) the original application code to size it correctly to be hashed with the secret authentication key, which is a different key than used earlier fro encryption. The result of the hash of the digested (or padded) application code with the authentication key is the Authentication MAC. That MAC gets downloaded with the encrypted code and seed number.

43. *Protecting Downloaded Firmware* with Symmetric Keys; Step2: Download, Decrypt, and Authenticate

The target user of protected firmware will download the encrypted code, seed number, and Authentication MAC.   To decrypt the code the seed is hashed with the secret decryption key stored in the ATSHA204 on the client embedded system. Being symmetric, this key has the same value as on the system the developer used for encryption. The result of that hashing process will be the decryption key, which of course matches the encryption key. The embedded system decrypts and recovers the application code using that key.

The process then moves to the authentication stage. The newly decrypted code is hashed (or padded) just like was done by the developer on the other side to prepare for hashing with the secret authentication key. Of course that key is the same as the authentication key the developer used, being symmetric. The result of that hash is the Client's Authentication MAC. It will be compared (using the CheckMAC command) to the Authentication MAC that was downloaded at the same time as the encrypted code and seed, and if the two Authentication MACs match then the downloaded and decrypted code is authenticated.


Download Protection (with Symmetric Keys)
Step2: Download, Decrypt & Authenticate
down loaded items
Decryption
Authentication
FLASH
Decrypt Code using recovered decryption key
AES Decryption
Decrypted Code
ENCRYPTED CODE
MCU
Decryption Key
SHA 256
Make digest of decrypted code
Load code if MACs compare
Hash seed w. secret key
Seed (number)
Digest of decrypted code
Compare MACs
Auth. MAC
SHA 256
SHA 256
Client Auth. MAC
Compare
Stored Secret Key for decryption
Secret Authentication Key
ATSHA204
Hash digest of Decrypted Code w. Auth. Key
© 2014 Atmel Confidential
CryptoAuthentication™ 2014
Atmel

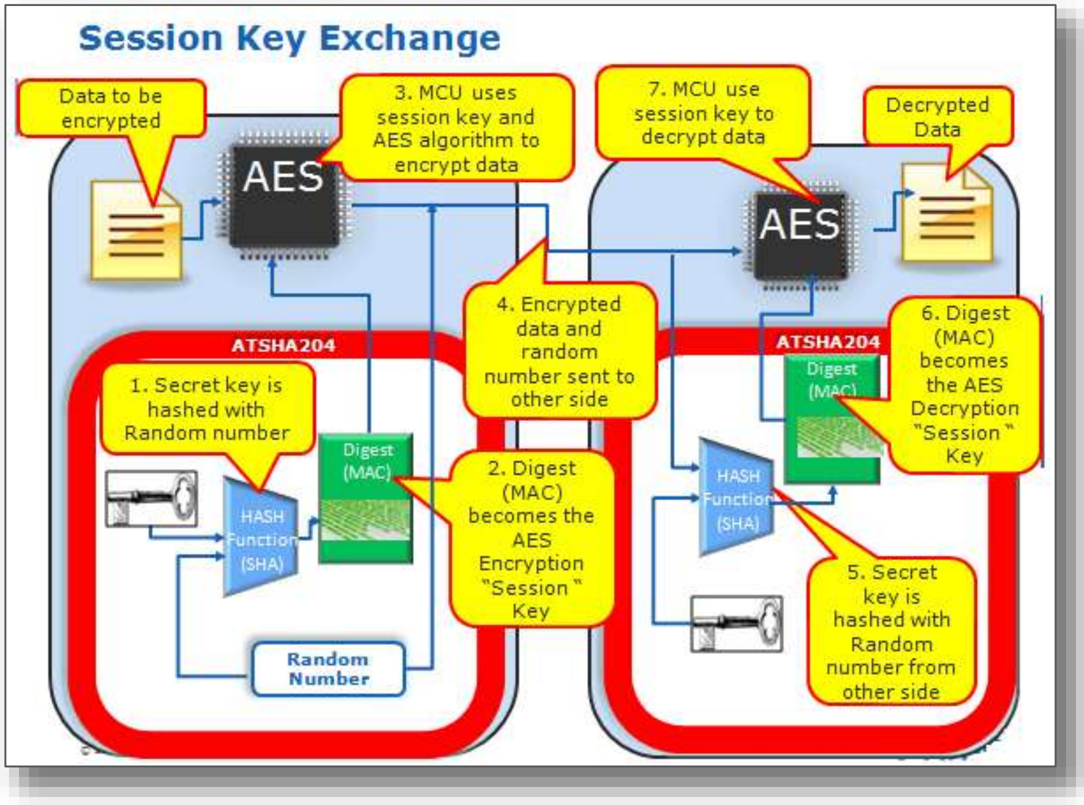
Benefits

- **Can have special downloads for each customer by tying use to specific authorized serial numbers.**
- **All downloads will be identical**
- **The user can post the download on the web because it will be encrypted**

44. *Symmetric Session Key Exchange* (with crypto devices on both sides)

The idea is to create a key that changes with each session. That is why is called a session key, of course. Hashing a key with the random number creates a 32 byte Message Authentication Code or "MAC". 16 bytes of that 32 byte (256 bit) MAC from the SHA204A becomes the AES session key that gets sent to the MCU to run the AES encryption algorithm over the data that is to be encrypted.

The newly encrypted data and the random number are then sent over to the other side so the data can be decrypted. Of course, to decrypt the message the same key that was used to encrypt it must be used in the decryption process. That is exactly why the random number is sent over, to recreate the session key from that random number and the key stored on the SHA204 on the decryption side. That is done by the random number being input to the SHA-256 hashing algorithm together with the key stored on the SHA204A. Because this is a symmetric operation the secret keys stored on the SHA204A devices on both sides are identical. So, when the same random number is hashed with the same secret key, then the 32 byte digest that results will be the same on the receiver (decrypting) side and on the sender (encrypting) side. Just like on the encrypting side, 16 bytes of the hash (i.e. MAC) represent the AES encryption/decryption key and will at this point be used to decrypt the message on the receiving side's MCU running the AES decryption algorithm.


Session Key Exchange
Data to be encrypted
3. MCU uses session key and AES algorithm to encrypt data
7. MCU use session key to decrypt data
Decrypted Data
AES
AES
4. Encrypted data and random number sent to other side
ATSHA204
ATSHA204
6. Digest (MAC) becomes the AES Decryption "Session" Key
1. Secret key is hashed with Random number
Digest (MAC)
Digest (MAC)
2. Digest (MAC) becomes the AES Encryption "Session" Key
HASH Function (SHA)
HASH Function (SHA)
5. Secret key is hashed with Random number from other side
Random Number

Benefits

- **Changes key for each session so keys are not used for very long which increases security**
- **Very simply yet secure methodology**
- **Easily implemented with ATSHA204A devices**

45. *Protecting Communication* between the Crypto device and MCU.

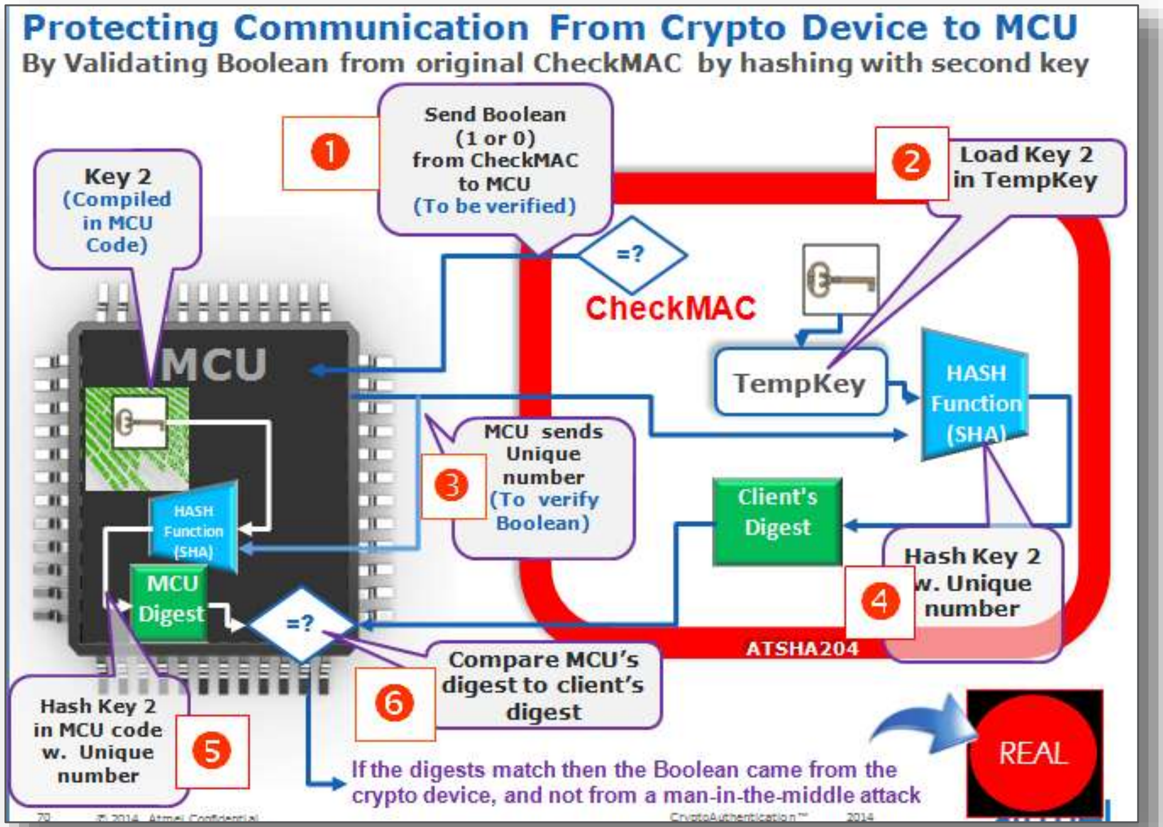
The question often comes up about how to ensure that the communication between the crypto device and the host MCU is not attacked by a man-in-the-middle.

This can be accomplished by performing an authentication on the result of CheckMAC function. The idea is to make sure that when the authentication device puts out the Boolean response from the Check MAC that response is not able to be tampered with.

From the diagram you can see that upon the CheckMac returning a response a second key (KEY 2) is loaded immediately into TempKey to that a second comparison can be made using a Unique number from the MCU that is hashed on the MCU and the authentication device (with KEY 2) and then compared.

All the devices have this method of protecting the single Boolean bit that comes from the authentication chip to the microprocessor.

It involves using a **second** key that is both stored in the CryptoAuthentication device and compiled into the code. After the successful completion of the 'Check' operation, the second secret is copied into the TempKey register. Then the micro sends over a unique number (call it time of day), which is then combined with that second secret using SHA and returned to the micro. The software on the micro does the same combination using the compiled secret to see if it agrees with the result from the authentication device. This is beneficial, because it means that you cannot just put a simple switch in the wire between the two and 'always send a 1'.


Protecting Communication From Crypto Device to MCU
By Validating Boolean from original CheckMAC by hashing with second key
1
Send Boolean (1 or 0) from CheckMAC to MCU (To be verified)
2
Load Key 2 in TempKey
Key 2 (Compiled in MCU Code)
=?
CheckMAC
MCU
TempKey
HASH Function (SHA)
3
MCU sends Unique number (To verify Boolean)
Client's Digest
HASH Function (SHA)
MCU Digest
=?
4
Hash Key 2 w. Unique number
ATSHA204
Hash Key 2 in MCU code w. Unique number
5
6
Compare MCU's digest to client's digest
If the digests match then the Boolean came from the crypto device, and not from a man-in-the-middle attack
REAL

Benefits

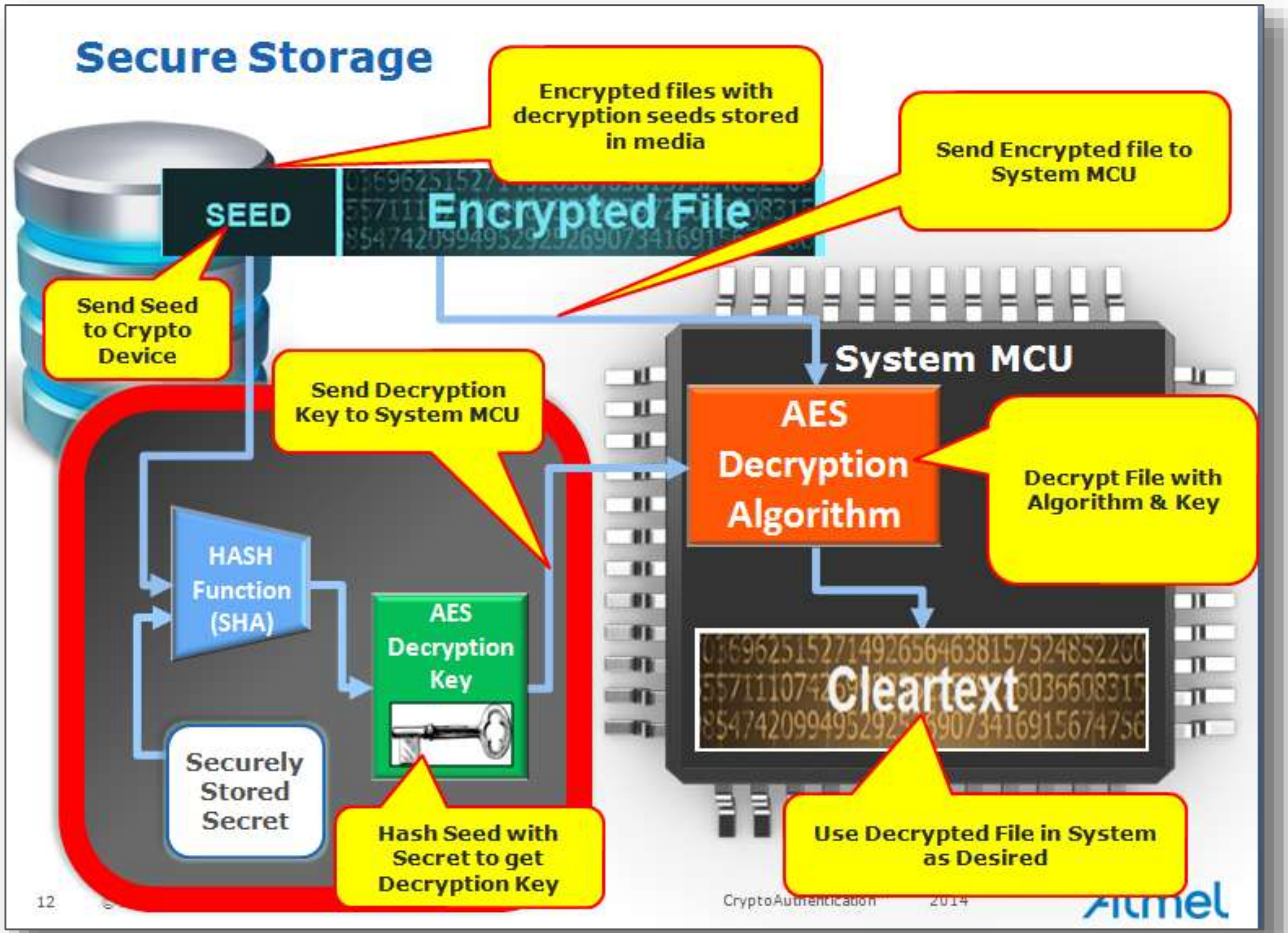
- **Provides complete security between the MCU and crypto device**
- **Capability built into every CryptoAuthentication device**
- **Simple to implement**

46. *Secure Storage* Using Encryption (Symmetric)

Hardware key storage devices can be used to encrypt files. The advantage is that the encryption-decryption key is securely stored in protected hardware as opposed to being stored in software that is inherently hackable (because all software is hackable).

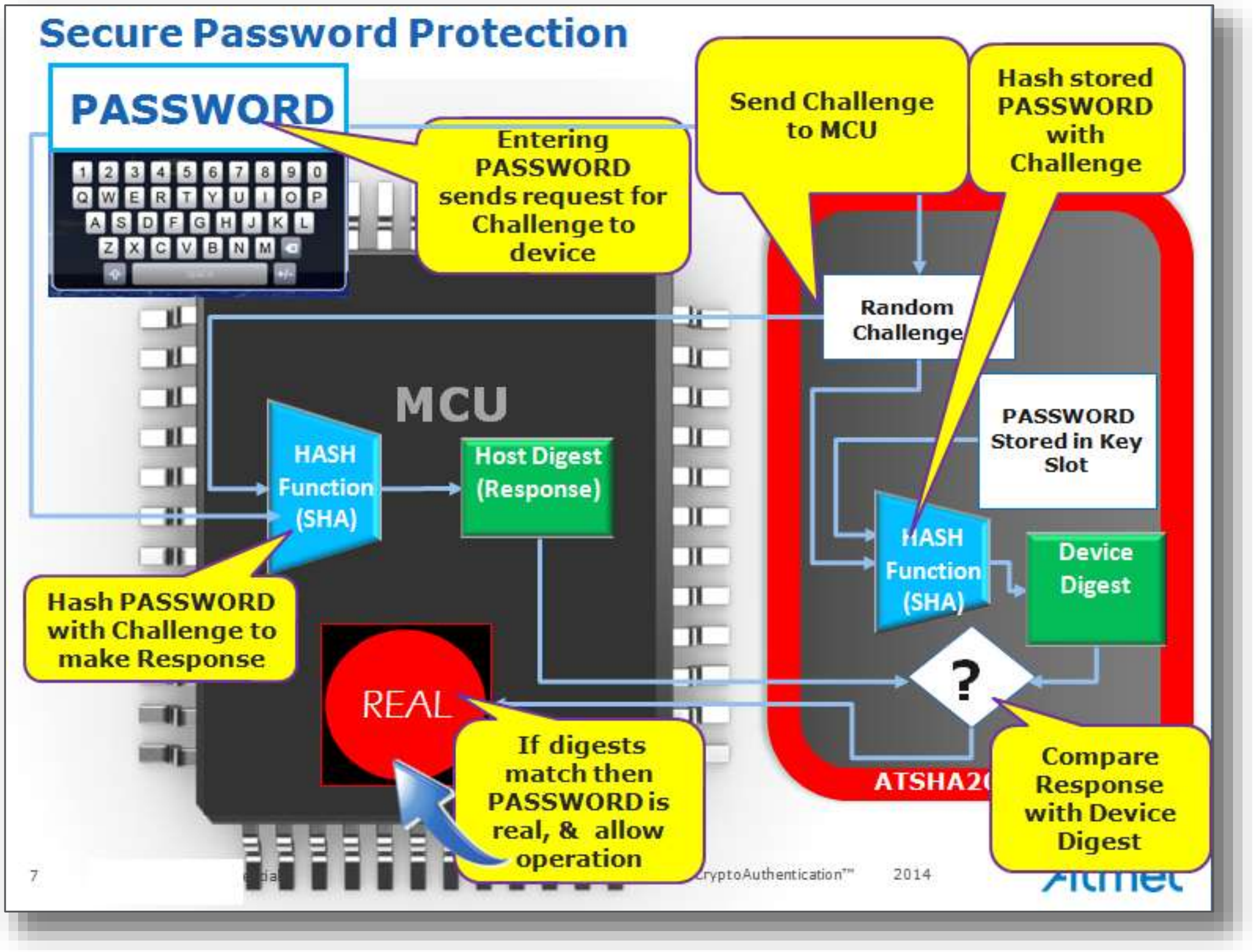
The encryption process in this example is AES (Advanced Encryption System) and the files were encrypted using a particular seed number that was hashed with the secret key stored securely at the encryption site. The hash of the secret key and seed created the AES decryption key used by the AES algorithm to encrypt the files.

To decrypt the encrypted files, the seed number is sent to the CryptoAuthentication device that securely stores the same secret. The crypto device hashes the seed and the secret to re-create the AES encryption-decryption key that is then used by the system MCU to decrypt the stored file into clear text

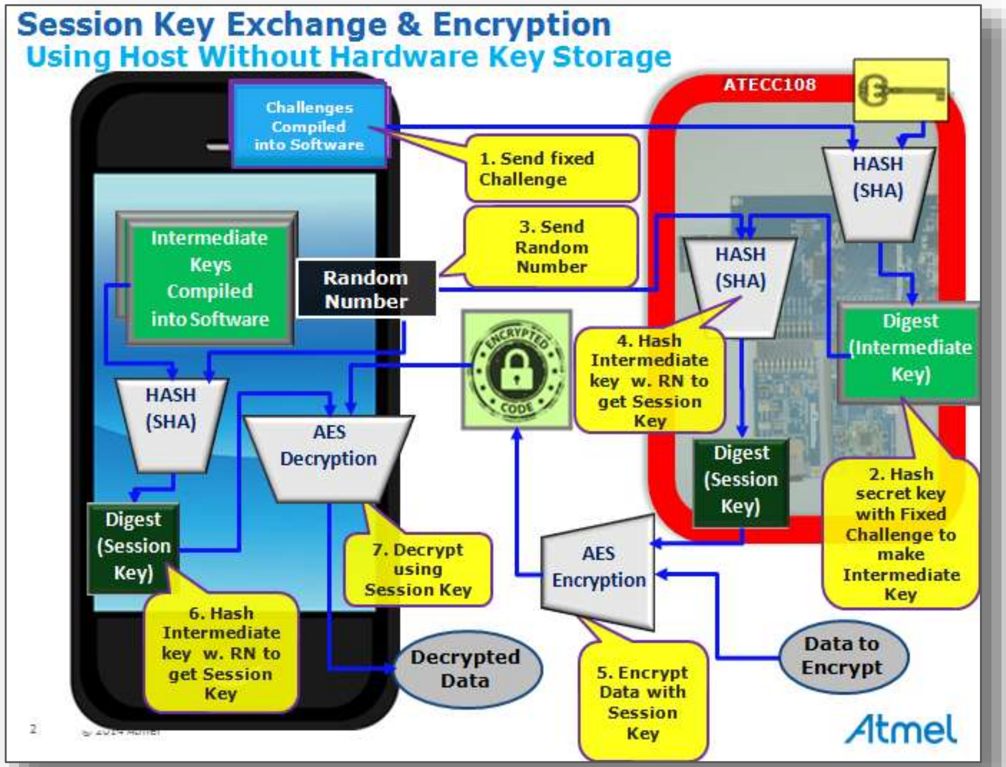

Secure Storage
Encrypted files with decryption seeds stored in media
Send Encrypted file to System MCU
SEED
Encrypted File
Send Seed to Crypto Device
System MCU
AES Decryption Algorithm
Send Decryption Key to System MCU
Decrypt File with Algorithm & Key
HASH Function (SHA)
AES Decryption Key
Securely Stored Secret
Cleartext
Hash Seed with Secret to get Decryption Key
Use Decrypted File in System as Desired
12
CryptoAuthentication
2014
Atmel

47. Secure Password Protection

To ensure that a password that for example is typed into a system is not intercepted, an authentication process is very useful. One example of how this can be done is to instruct that the CryptoAuthentication device sends a random challenge to the system MCU at the time the password is being entered. That random challenge then gets hashed with the password to create a response digest. The response digest gets sent to the crypto device which also hashed the same random challenge and the password with is stored in a key slot on the device. The stored password and random challenge number are hashed to create a digest which is compared to the response digest made on the MCU, If the digests match then the password is considered


Secure Password Protection
PASSWORD
Entering PASSWORD sends request for Challenge to device
Send Challenge to MCU
Hash stored PASSWORD with Challenge
Random Challenge
PASSWORD Stored in Key Slot
MCU
HASH Function (SHA)
Host Digest (Response)
HASH Function (SHA)
Device Digest
?
Hash PASSWORD with Challenge to make Response
REAL
If digests match then PASSWORD is real, & allow operation
Compare Response with Device Digest
ATSHA2
7
CryptoAuthentication™
2014

48. *Confidentiality* using Symmetric Session Key Exchange


Session Key Exchange & Encryption
Using Host Without Hardware Key Storage
Challenges Compiled into Software
1. Send fixed Challenge
ATECC108
HASH (SHA)
3. Send Random Number
Intermediate Keys Compiled into Software
Random Number
HASH (SHA)
Digest (Intermediate Key)
4. Hash Intermediate key w. RN to get Session Key
HASH (SHA)
AES Decryption
Digest (Session Key)
2. Hash secret key with Fixed Challenge to make Intermediate Key
Digest (Session Key)
7. Decrypt using Session Key
AES Encryption
6. Hash Intermediate key w. RN to get Session Key
Decrypted Data
5. Encrypt Data with Session Key
Data to Encrypt
Atmel

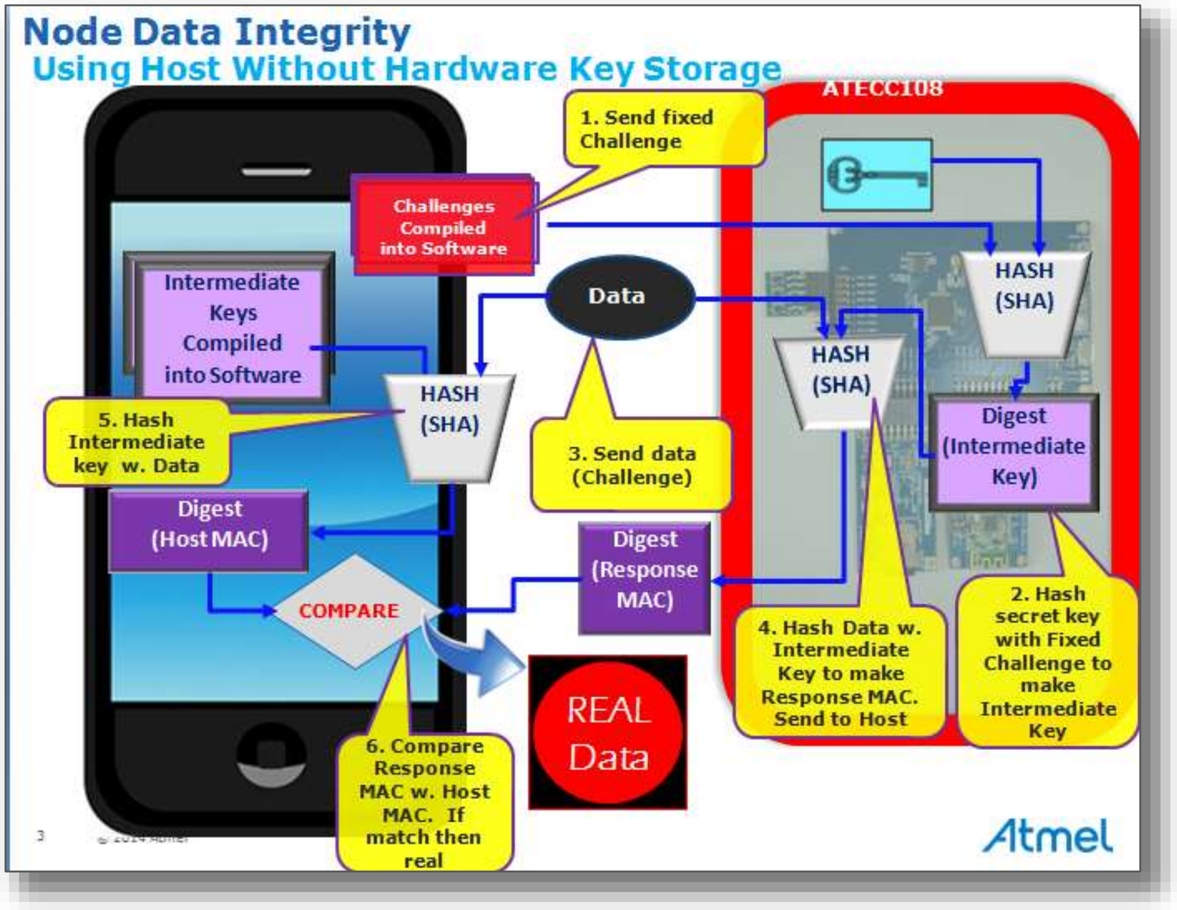
Confidentiality is making sure that the message cannot be seen by an unintended party. The method used is encryption and decryption based upon a selected algorithm, which in this example is Advanced Encryption System (AES). To perform encryption and decryption, both sides need the same encryption/decryption key which is input to the algorithm on each side. The trick is to produce the key in each side in a way that does not allow anyone else to obtain it. An excellent way to do that is to create a new key for each communication session by sending information that only the intended parties can use to create the keys. The fact that a new key is created for each session increases security greatly. This method is not surprisingly called **session key exchange**. This is what we will now describe.

The example here is symmetric key exchange, meaning that the session keys on each side will be the same. To exchange symmetric session keys in platforms where it is not possible or desirable to securely store a secret key in the host, a method that preloads challenges and intermediate keys in the host is recommended. This can be called a **"fixed challenge"** method.

The process starts with a challenge number that was loaded into the host's software being sent to the remote node (Step 1). The node has a secret key stored there (in the ATSHA204 or ATECC108 device). This secret key is hashed with the challenge send over from the host to create a digest (Step 2). Let's call that the encryption intermediate key. The host side has that very same intermediate key already loaded in its software. The host's intermediate key was created using the same hash algorithm with the same challenge and same secret key as on the client. That process occurs on tester in the factory. The tester securely stores the secret key and the only the calculated intermediate key is released. The secret key stays in the factory and stays secret. Once the hashes are run on the tester to make the intermediate keys, the challenges and corresponding intermediate keys are made available to be compiled into the host's software. Now both sides have the same intermediate key, which will be used to create the session keys on each side.

To create the session keys, the host sends a random number to the node (Step 3), and the node hashes that random number with the intermediate key that was created in Step 2 to form the Encryption Session Key (Step 4). The session key can now be input to the encryption algorithm to encrypt the original message, which gets sent to the host (Step 5).The host hashes the intermediate key in its software with the same random number it sent to the node to form the identical session key as on the Node called the Decryption session Key (Step 6). That key is used to decrypt the message that was sent over from the Node. That completes the process.

49. *Node Data Integrity* Using Symmetric Intermediate Keys


Node Data Integrity
Using Host Without Hardware Key Storage
ATECC108
1. Send fixed Challenge
Challenges Compiled into Software
Data
HASH (SHA)
Intermediate Keys Compiled into Software
HASH (SHA)
HASH (SHA)
Digest (Intermediate Key)
5. Hash Intermediate key w. Data
3. Send data (Challenge)
Digest (Host MAC)
Digest (Response MAC)
COMPARE
4. Hash Data w. Intermediate Key to make Response MAC. Send to Host
2. Hash secret key with Fixed Challenge to make Intermediate Key
REAL Data
6. Compare Response MAC w. Host MAC. If match then real
Atmel

To provide node data integrity in a platform where it is not possible or desirable to securely store a secret key on the host, a fixed challenge method using an intermediate key is recommended. A fixed challenge is a known number that has previously been loaded into the host's software (as opposed to a random challenge where the challenge is a random number). To start the process the challenge number is sent by the host to the remote node (Step 1) to be hashed with a secret Integrity key stored there (Step 2). The integrity key is different from the other secret keys stored in the node and used for authentication and confidentiality. The digest of that hashing of the secret key and challenge becomes the Intermediate Integrity Key. The same Intermediate Integrity Key is already stored in software on the host. It was previously created in the factory using the same challenge number which was hashed in the factory with the same secret key securely stored on the tester in the factory using the same hashing algorithm. The integrity key once created in the factory was made available to be compiled into the host's software along with the corresponding challenge number from which it was created.

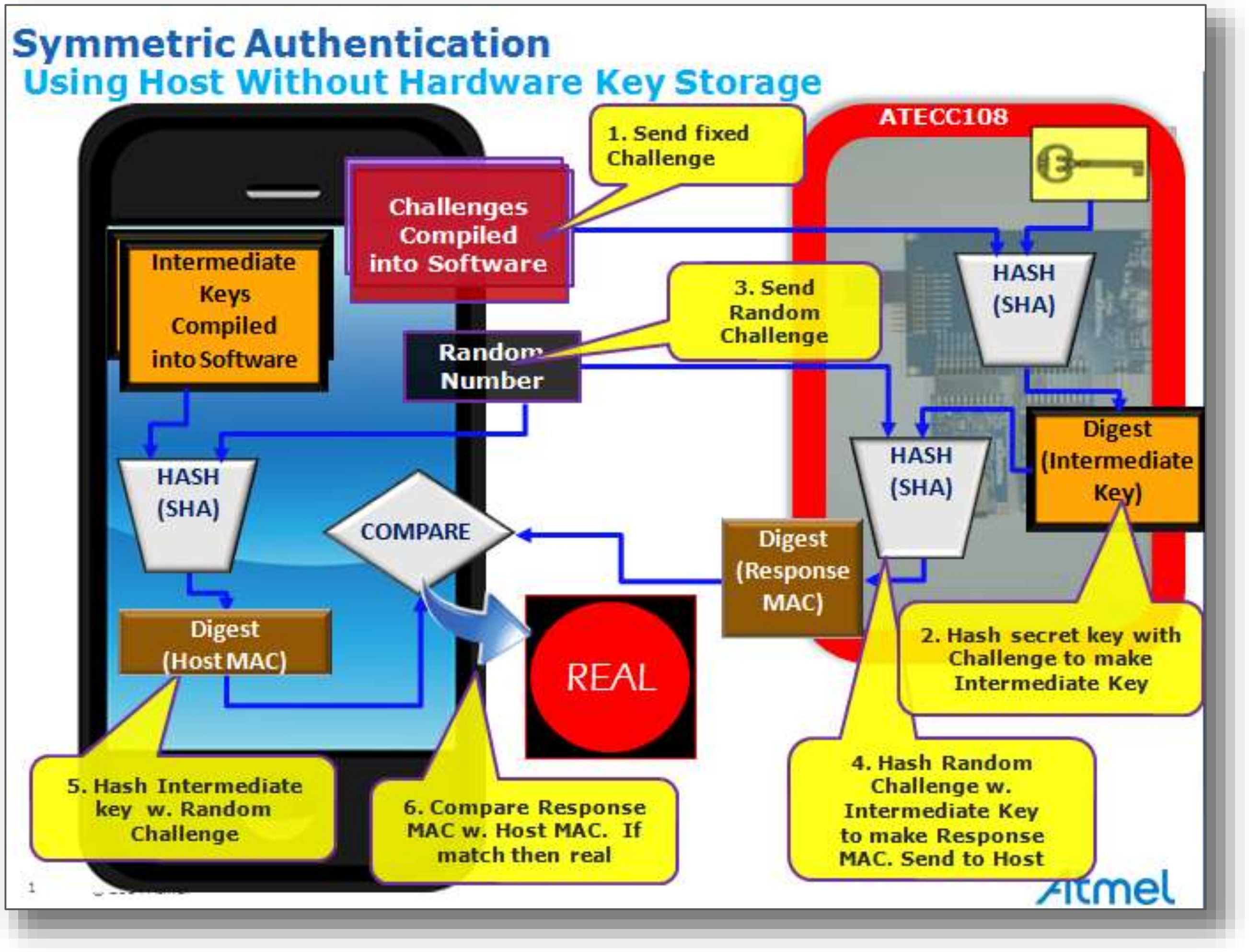
Moving back to the Node, the original message gets hashed with the intermediate key on the Node to create a Message Authentication Code (MAC), which is called the Message MAC (Step 4). But first the message needs to be prepared for that step by being sized correctly. That is done either by digesting the message or padding it, depending on the size of the original message (Step 3). 32 bytes is the size of the input to the SHA 256 hashing step. The Message MAC is then added to the original message and that combined code is encrypted and sent to the host (Step 5).

When the host receives the encrypted code it decrypts it to retrieve the original message and the Message MAC (Step 6). The host will soon use the Decrypted Message MAC to check for authenticity. It does that by hashing the intermediate key stored on the host with the decrypted message, which is prepared for hashing as done on the other side (step 7). That hash creates the Host Message MAC which is then compared with the Decrypted Message MAC (step 8). If they match then the data integrity of the message is verified.

50. *Node Authentication* Using Symmetric Intermediate Keys

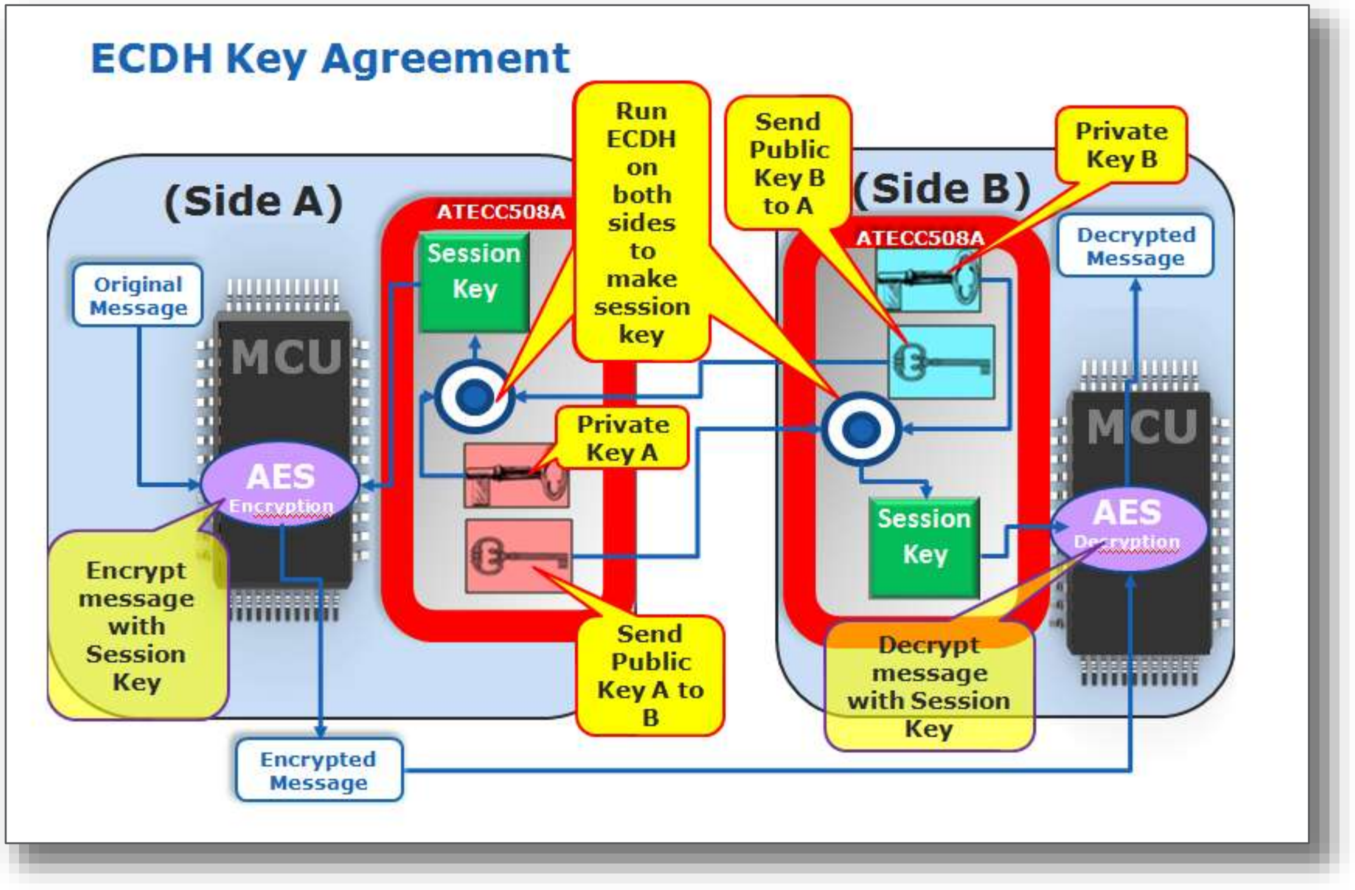
To provide node authentication in a platform where it is not possible or desirable to securely store a secret key on the host, a method called fixed challenge-response using an intermediate key is recommended. A fixed challenge is a number that has previously been loaded into the host's software. To start the process that challenge number is sent to the remote node (Step 1) to be hashed with a secret authentication key stored there (Step 2). That key is different from the other secret keys stored on the ATSHA204 that are used for confidentiality and data integrity. The digest of that hashing process is the intermediate authentication key. That same challenge number was previously hashed in the factory with the same secret key securely stored there (on the tester) and the same hashing algorithm to create that very same intermediate key. That intermediate key was then compiled into the host's software along with the challenge.

The host then sends a random number to the node (Step 3). The host hashes its intermediate key with the random number to create a digest called the Node Response (Step 4), which is then sent to the host. On the host, the same random number is hashed with the intermediate key stored on the host (Step 5) to create a digest called the Host Response. The host' response and node responses are compared, and if those digests match then the authenticity of the node is verified (Step 6).


Symmetric Authentication
Using Host Without Hardware Key Storage
ATECC108
1. Send fixed Challenge
Challenges Compiled into Software
Intermediate Keys Compiled into Software
3. Send Random Challenge
HASH (SHA)
Random Number
HASH (SHA)
Digest (Intermediate Key)
HASH (SHA)
COMPARE
Digest (Response MAC)
Digest (Host MAC)
REAL
2. Hash secret key with Challenge to make Intermediate Key
5. Hash Intermediate key w. Random Challenge
6. Compare Response MAC w. Host MAC. If match then real
4. Hash Random Challenge w. Intermediate Key to make Response MAC. Send to Host
Atmel

51. *Confidentiality* using ECDH Key Agreement (with ATECC508A)

Confidentiality is making sure that the message cannot be seen by an unintended party. The method used is encryption and decryption based upon a selected algorithm, which in this example is Advanced Encryption System (AES). To perform encryption and decryption, both sides need the same encryption/decryption key which is input to the algorithm on each side. The trick is to produce the key in each side in a way that does not allow anyone else to obtain it. An excellent way to do that is to create a new key for each communication session by sending information that only the intended parties can use to create the keys. The fact that a new key is created for each session increases security greatly. This method is not surprisingly called **session key exchange**. This is what we will now describe.



The example here is ECDH key exchange which stands for Elliptic Curve Diffie-Helman. The ECDH mathematics called point-multiply (depicted by the dot in the diagram) works such that the private key of side A point multiplied by the public key of side B is exactly equal to the private key of side B point-multiplied by the public key of side A. The key point is that knowing the public key of either side does not allow anyone else other than the intended parties to obtain the shared session key. (The mathematical magic of elliptic curve point-multiply is explained elsewhere.) Once the encrypting side (which can be either side but is defined as side A in this example, get the session key that can be used by the microprocessor to encrypt the message using AES (in this example. The encrypted message is then sent to the other side (Side B) which also now has the same session key due to the result of the ECDH calculation don on Side B's private key and Side A's public key. The micro then uses the session key to decrypt the encrypted message. That completes the key agreement and encryption/decryption process.

To obtain data integrity an alternative model of AES can be employed such as AES-CCM or AES-GCM. To obtain authenticity the ECDSA operation in the ATECC508A can be run. Therefore all three of the pillars of security are possible using the ATECC508A, namely Confidentiality, Integrity, and Authentication (which ironically is sometimes referred to as "CIA").

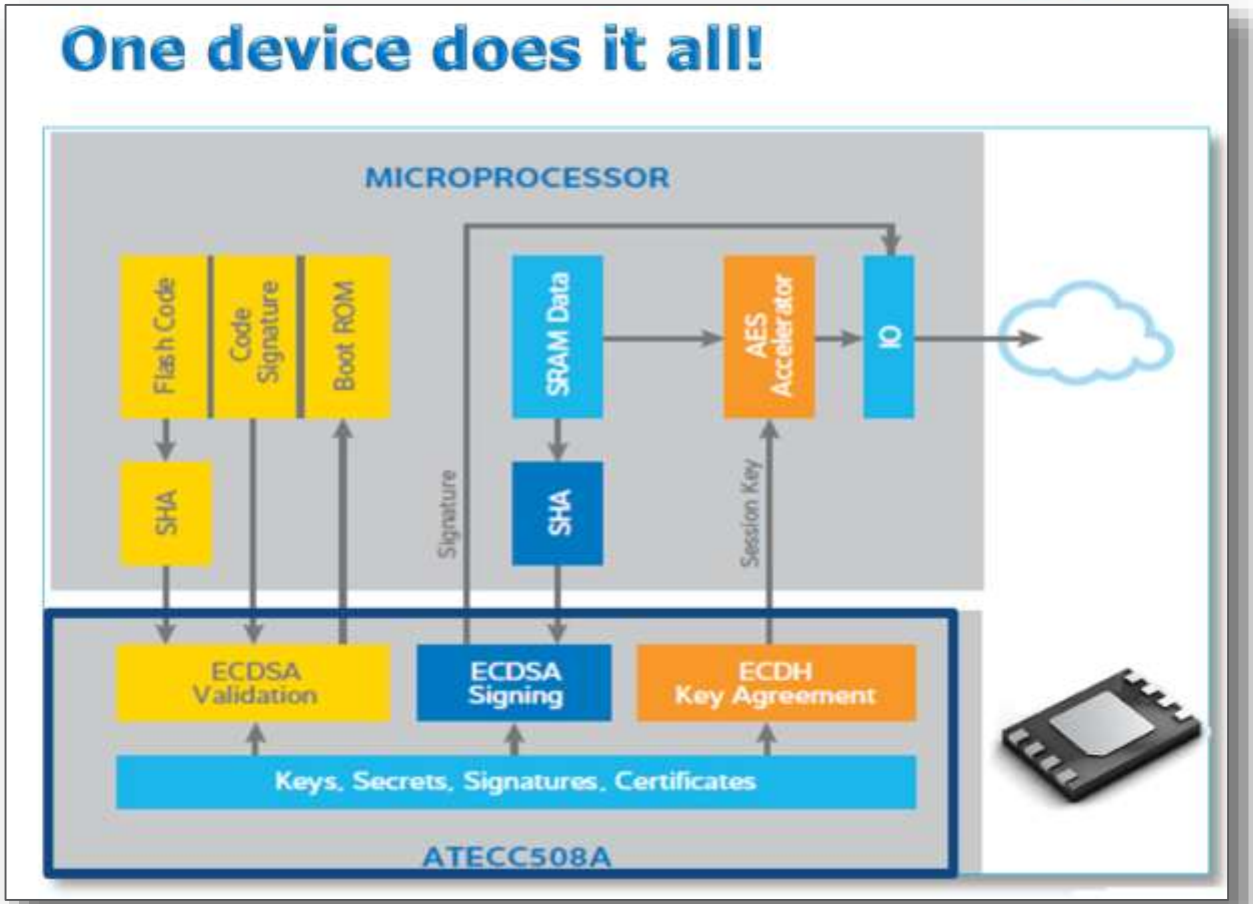
52. *ATECC508A* Provides Confidentiality, Integrity, and Authenticity.

The ATECC508A stores keys, secrets, and certificates on-chip in protected hardware, which is the strongest form of storage. The ATECC508A provides ECDSA (P256 curve) to implement authentication via the sign-verify process.

The device also runs ECDH (Elliptic Curve Diffie-Helman) Key Agreement capability to implement confidentiality (via encryption/decryption). The ATECC508A is a perfect companion for microprocessors running encryption algorithms such as AES, because of the built in ECDH key agreement functionality, which easily provides the microprocessor with the encryption/decryption keys on each side of the process using simple command calls.

The ATECC508A can very simply implement secure boot using the ECDSA validation process (the yellow blocks in the diagram) ensuring that only the correct (i.e. validated) code is loaded at boot time.

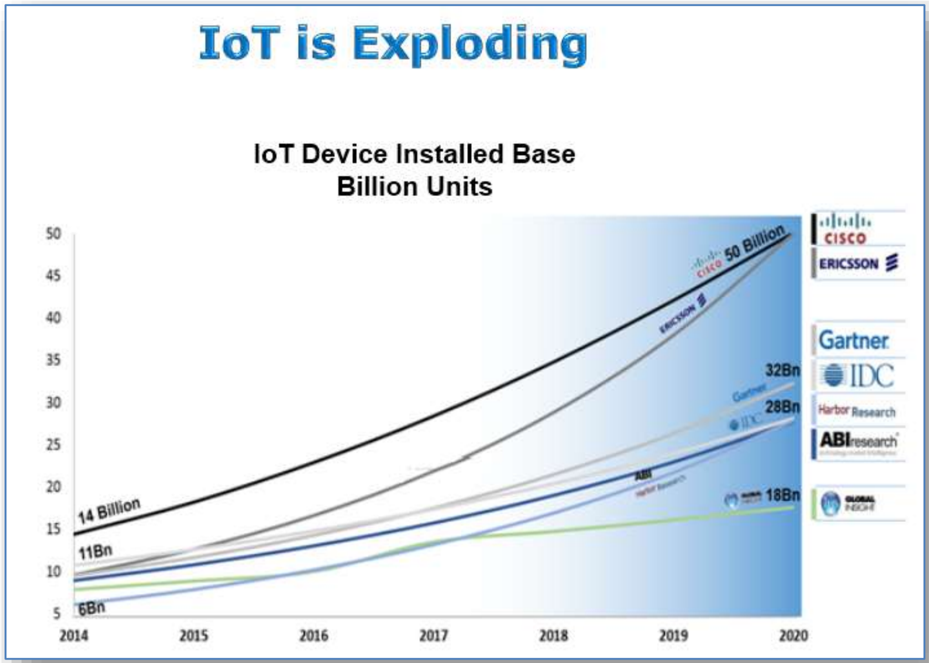
In systems where ECDH key agreement is used with microprocessors running AES, data Integrity can be easily provided using various mechanisms such as AES-CCM and AES-GCM.

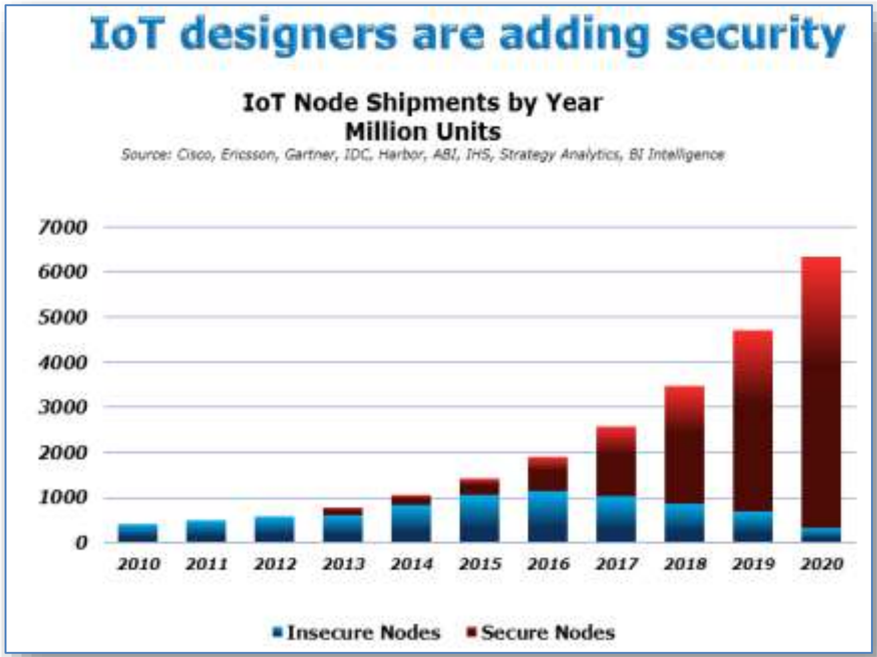


Therefore, all three of the pillars of security are supported by the ATECC508A; namely, confidentiality, integrity, and authentication (sometimes called “CIA”), making this a very elegant and robust security solution.

53. Security and the IoT

Security matters in the IoT because users must trust that the nodes are who they say are (i.e. are authentic). Additionally, confidentiality of the data is important to keep unauthorized third parties from getting the data and misusing it. Also, without data integrity mechanisms there is no way to ensure that data have not been tampered with or corrupted. All three of these matter...A lot. Without security the IoT cannot be trusted. Fortunately, the ATECC508A addresses all of the pillars of security. Crypto engines like the ATECC508 with secure hardware-based key storage will help catalyze the growth of the IoT. As you can see in the charts, the growth of the IoT will be astonishing, but without real security the IoT will remain simply a toy. Clearly, security matters in the IoT market…and everywhere else.





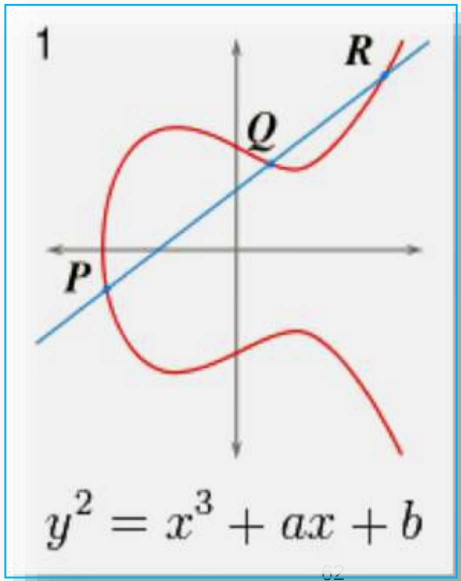
54. *The Mathematical Magic* of ECDH Key Agreement

What if you and I want to exchange encrypted messages? Encryption is essentially scrambling a message so only the intended reader can see it after they unscramble it. By definition, scrambling and unscrambling are inverse (i.e. reversible) processes. Doing and undoing mathematical operations in a secret way, which outside parties cannot understand or see, is the basis of encryption/decryption. Julius Caesar used encryption to communicate privately. The act of shifting the alphabet by a specific number of places is still called the Caesar cipher. Note that the number of places is kept secret and acts as the key.

A modern-day encryption key is a number that is used by an encryption algorithm, such as AES (Advanced Encryption Standard) and others, to encode a message so no one other than the intended reader can see it. Only the intended parties are supposed to have the secret key. The interaction between a key and the algorithm is of fundamental importance in cryptography of all types. That interaction is where the magic happens. An algorithm is simply the formula that tells the processor the exact, step-by-step mathematical functions to perform and the order of those functions. The algorithm is where the magical mathematical spells are kept, but those are not kept secret in modern practice. The key is used with the algorithm to create secrecy.

For example, the magic formula of the AES algorithm is a substitution-permutation network process, meaning that AES uses a series of mathematical operations done upon the message to be encrypted and the cryptographic key (crypto people call the unencrypted message “plaintext“). How that works is that the output of one round of calculations done on the plaintext is substituted by another block of bits and then the output of that is changed (i.e. permutated) by another block of bits and then it happens over and over, again and again. This round-after-round of operations changes the coded text in a very confused manor, which is the whole idea. Decryption is exactly as it sounds, simply reversing the entire process.

That description, although in actual fact very cursory, is probably TMI here, but the point is that highly sophisticated mathematical cryptographic algorithms that have been tested and proven to be difficult to attack are available to everyone. If a secret key is kept secret, the message processed with that algorithm will be secret from unintended parties. This is called Kerckhoffs' principle after it author Auguste Kerchoffs who was a Dutch professor, linguist, and cryptographer in the 19th century. His principle could be paraphrased as, “The key to encryption is the key.” This powerful concept is the heart of modern cryptography. What it says is that you need both the mathematical magic and secret keys for strong cryptography.



Another way to look at is that the enemy can know the formula, but it does him or her no good unless they know the secret key. That is, by the way, why it is so darn important to keep the secret key secret. Getting the key is what many attackers try to do by using a wide variety of innovative attacks that typically take advantage of software bugs. So, the best way to keep the secret is to store the key in secure hardware that can protect if from attacks. Software storage of keys is just not as strong as hardware storage. Bugs are endemic, no matter how hard the coders try to eliminate them. Hardware key storage trumping software is another fundamental point worth remembering. So now that we have a good algorithm (e.g. AES) and a secret key we can start encrypting to gain confidentiality.

The Mathematical Magic of ECDH Key Agreement (Continued)

Key Agreement

In order for encryption on the sender's side and decryption on the receiver's side, both sides must agree to have the same key. That agreement can happen in advance, but that is not practical in many situations. As a result, there needs to be a way to exchange the key during the session where the encrypted message is to be sent. Another powerful cryptographic algorithm will be used to do just that.

There is a process called ECDH key agreement, which is a way to send the secret key without either of the sides actually having to meet each other. ECDH uses a different type of algorithm from AES that is called "EC" to send the secret key from one side to the other. EC stands for elliptic curve, which literally refers to a curve described by an elliptic equation.

A certain set of elliptic curves (defined by the constants in the equation) have the property that given two points on the curve (P and Q) there is a third point, P+Q, on the curve that displays the properties of commutivity, associativity, identity, and inverses when applying elliptic curve point multiplication. Point-multiplication is the operation of successively adding a point along an elliptic curve to itself repeatedly. Just for fun the shape of such an elliptic curve is shown in the diagram above.

The thing that makes this all work is that EC point-multiplication is doable, but the inverse operation is not doable. Cryptographers call this a one-way or trap door function. (Trap doors go only one way, see?) In regular math, with simple algebra if you know the values of A and A times B you can find the value of B very easily. With Elliptic curve point-multiply if you know A and A point-multiplied by B you cannot figure out what B is. That is the magic.

That irreversibility and the fact that A point-multiplied by B is equal to B point-multiplied by A (i.e. commutative) are what makes this a superb encryption algorithm, especially for use in key exchange.

To best explain key agreement with ECDH, let's say that everyone agrees in advance on a number called **G**, which is a constant point on the Elliptical curve. Now we will do some point-multiply math. Let's call the sender's private key **PrivKeySend.** (Note that each party can be a sender or receiver, but for this purpose we will name one the sender and the other the receiver just to be different from using the typical Alice and Bob nomenclature used by most crpyto books.) Each private key has a mathematically related and unique public key that is calculated using the elliptic curve equation. Uniqueness is another reason why elliptic curves are used. If we point-multiply the number **G** by **PrivKeySend** we get **PubKeySend**. Let's do the same thing for the receiver who has a different private key called **PrivKeyReceive** and point-multiply that private key by the same number **G** to get the receiver's public key called **PubKeyReceive**.   The sender and receiver can then exchange their public keys with each other on any network since the public keys do not need to be kept secret. Even an unsecured email is fine.

Now, the sender and receiver can make computations using their respective private keys (which they are securely hiding and will never share) and the public key from the other side. Here is where the commutative law of point-multiply will work its magic.

The sender point-multiplies the public key from the other side by his or her stored private key.

This is equates to:

PubKeyReceive point-multiplied by **PrivKeySend** which = **G** point-multiplied by **PrivKeyReceive** point-multiplied by **PrivKeySend**

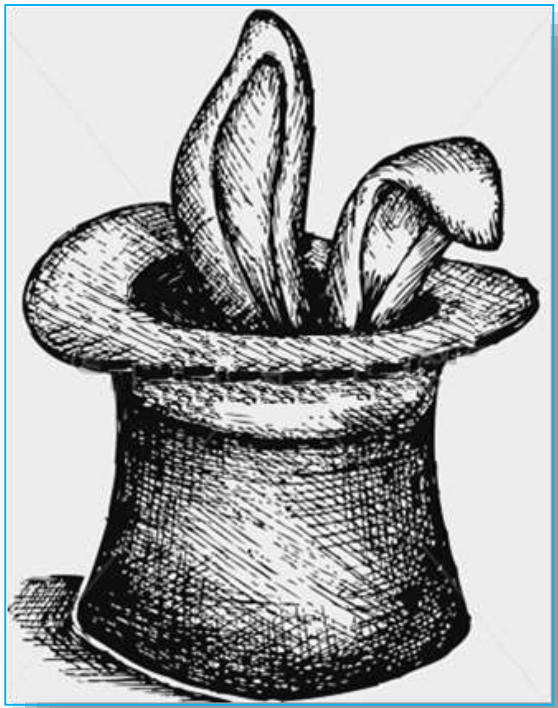
The receiver does the same thing using his or her private key and the public key just received.

This equates to:

PubKeySend point-multiplied by **PrivKeyReceive** = **G** point-multiplied by **PrivKeySend** point-multiplied by **PrivKeyReceive**.

Because point-multiply is commutative these equations have the same value!

The Mathematical Magic of ECDH Key Agreement (Continued)



And, the rabbit comes out of the hat: The sender and receiver now have the exact same value, which can now be used as the new encryption key for AES in their possession. No one else can get that value. That is because someone else would need to have one of the private keys, which they cannot get.

This calculated value can now be used by the AES algorithm to encrypt and decrypt messages. Pretty cool, isn't it?

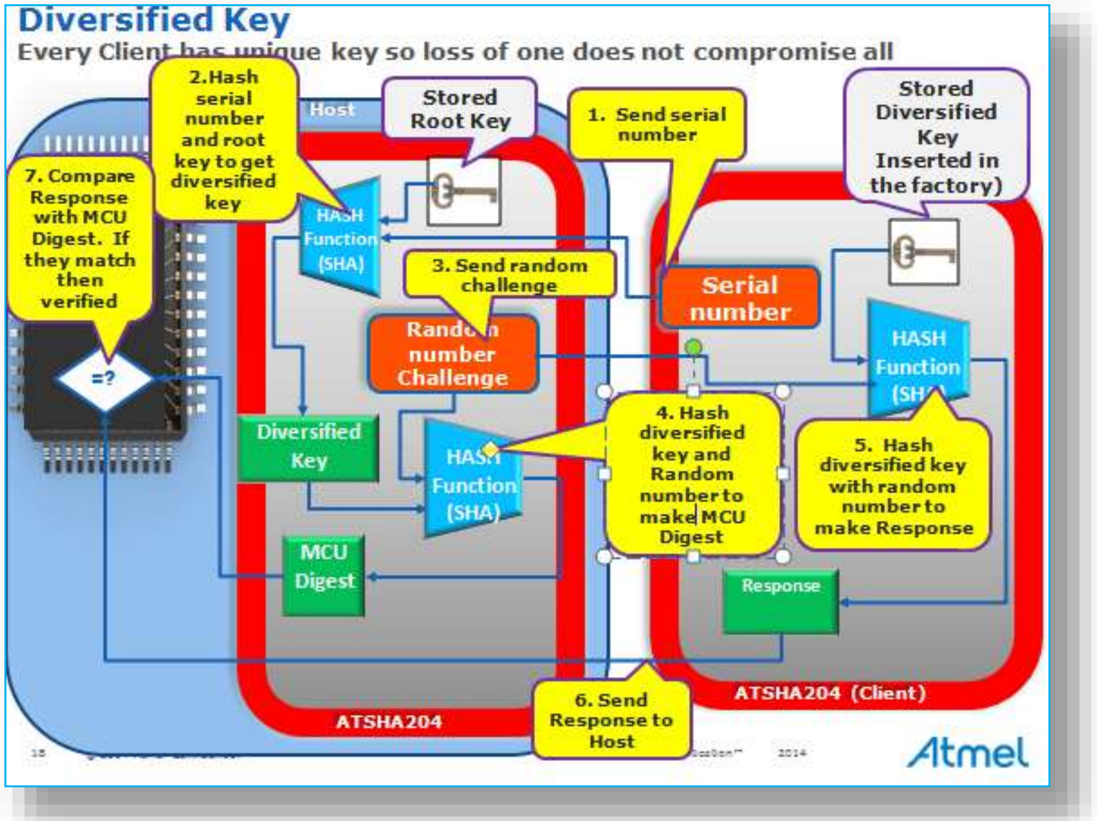
Below is a link to a wonderful video explaining the modular mathematics and discrete logarithm problem that creates the one-way, trapdoor function used in Diffie-Hellman key exchange. (Oh yeah, the "DH" in ECDH stands for Diffie-Hellman who were two of the inventors of this process.)

http://www.youtube.com/watch?v=3QnD2c4Xovk

55. *Using diversified keys* to increase security

An excellent way to increase security in a symmetric operation is to use something called diversified keys. This is also called a Key Derivation Function (or KDF). What that means is that the key that is in the host, called the root key, is not shared as such with the clients. Instead each client gets a derivative of that root key based upon a unique number assigned to that particular client, such as its serial number. That serial number is hashed with the root key in the factory and stored in the client device and it is known as the “diversified key”. It is called diversified since each client’s key will be different from that is other clients (i.e. they are diverse).

To authenticate with a diversified key, the host must be able to create the diversified key just like it was created in the factory. Therefore, it will need the serial number from the client. So, the client sends the serial number to the host MCU and the host hashes the serial number with the stored secret root key. That hash value (digest) should then be the same as the client’s diversified key. The next step is to use a random number challenge-response operation to verify that the keys are the same, and thus that the client is real. To do so, the host sends a random number challenge to the client which then hashes that number with the stored diversified key (that was inserted in the factory). The digest of the key and challenge is the Response, and that is sent to the host. The host hashes the same random number with the diversified key it created from the client’s serial number and stored root key. It compares that digest with the response to see if they match. If they do then the client is considered as being real.


Diversified Key
Every Client has unique key so loss of one does not compromise all
2.Hash serial number and root key to get diversified key
Host
Stored Root Key
1. Send serial number
Stored Diversified Key Inserted in the factory)
7. Compare Response with MCU Digest. If they match then verified
HASH Function (SHA)
3. Send random challenge
Serial number
Random number Challenge
=?
Diversified Key
HASH Function (SHA)
4. Hash diversified key and Random number to make MCU Digest
5. Hash diversified key with random number to make Response
HASH Function (SHA)
MCU Digest
Response
ATSHA204
6. Send Response to Host
ATSHA204 (Client)
Atmel

Benefits

- **All clients will have a different key so the loss of one does not affect any others**
- **Simple operation to improve security substantially**
- **Users can use this technique to easily create client blacklists**

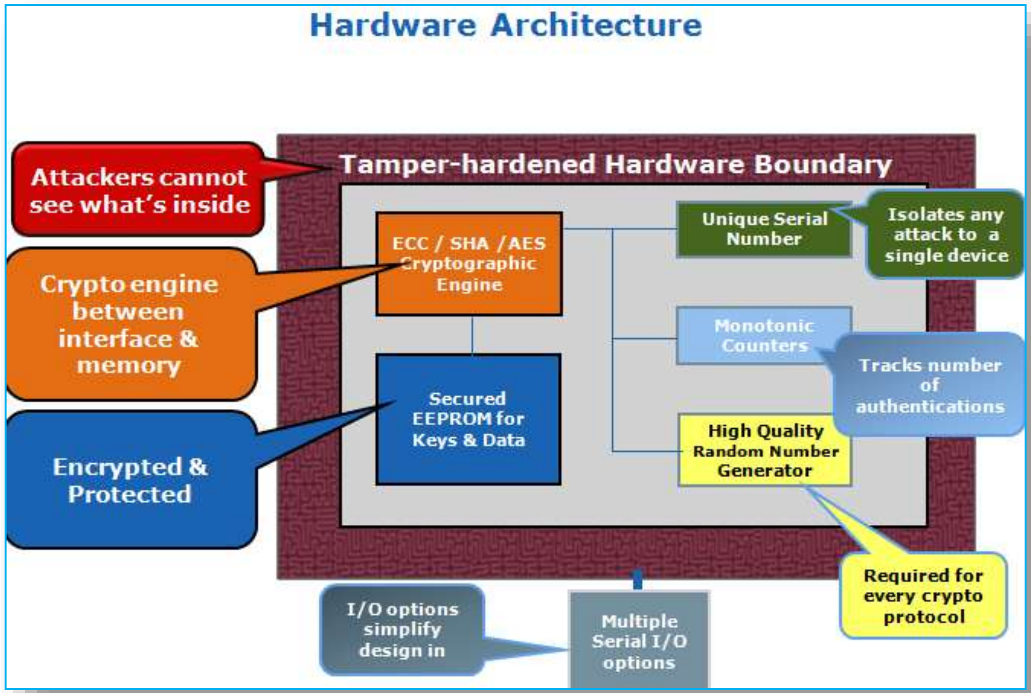
56. *Innovations* that make Atmel secure key storage devices robust.

Looking at the red border in the diagram you can see a tamper-hardened hardware boundary.

This boundary contains defense mechanisms such as shields, regulators, clock generators, and other counter-measures so attackers cannot see what is inside. You can see in the lower picture the structure of the serpentine metallic shield that covers the _entire_ chip. This shield is integral to the active circuitry so even if the device is deprocessed, nothing will be revealed. So, attackers cannot see what is inside, and they cannot attack what they cannot see.

All the devices employ the same basic architecture of secure EEPROM for keys & data with the crypto engine standing between the interface and memory.

A one-way counter tracks how many times the device authenticates. Even when the power is turned off, the count remains the same.   All the devices include internal high quality hardware Random Number Generator, which is required for every cryptographic protocol. In addition, all the products include a guaranteed unique serial number for identification and key diversification.

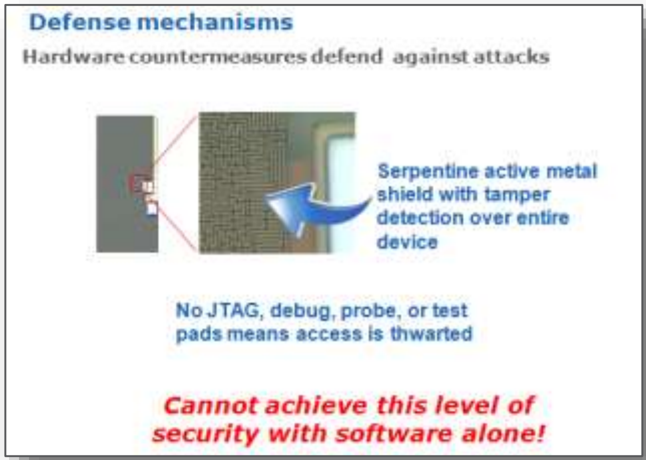

Hardware Architecture
Tamper-hardened Hardware Boundary
Attackers cannot see what's inside
Crypto engine between interface & memory
Encrypted & Protected
ECC / SHA /AES Cryptographic Engine
Secured EEPROM for Keys & Data
Unique Serial Number
Isolates any attack to a single device
Monotonic Counters
Tracks number of authentications
High Quality Random Number Generator
Required for every crypto protocol
I/O options simplify design in
Multiple Serial I/O options

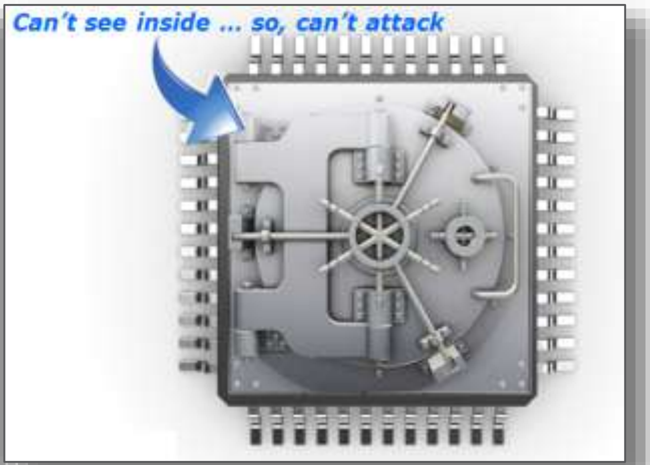
The memories are all internally encrypted, and an array of tamper-detection schemes is in force. JTAG is not used, and there are no debug, probe, or test points.   Such design methodologies add up to a ultra-strong defense against micro probing, timing, emissions and other types of attacks.

Simply put: "It is not possible to achieve this level of security with software alone."

Benefits

- **Multiple levels of hardware security**
- **Cannot achieve this level of security with software alone!**
- **Guards against numerous styles of attacks**


Defense mechanisms
Hardware countermeasures defend against attacks
Serpentine active metal shield with tamper detection over entire device
No JTAG, debug, probe, or test pads means access is thwarted
Cannot achieve this level of security with software alone!



57. Consumption Tracking

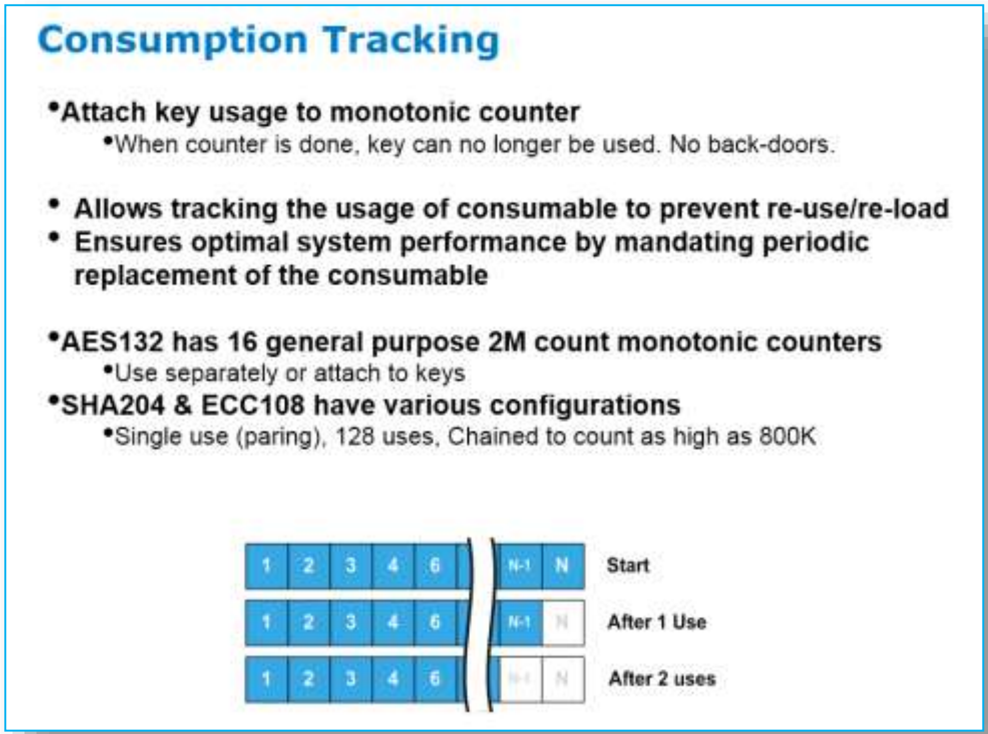
The consumption tracking feature on CryptoAuthentication devices offers an easy way to control various use cases such as the number of times a product can be used, which can help to ensure the quality of products and systems.   By counting the number of uses designers can ensure that the accessory such as a printer ink tank, for example, will not operate when the ink is too low for a high quality image.

Another application of the counter is called “pairing restriction” which is a one-to-one match between one host and one client being allowed, which adds more security benefits. Once a client is used with a specific host it can then not be used on another host.

CryptoAuthentication devices can also attach the counter to cryptographic keys. This is done be using the counter as an input to a hash function with a stored key to create a MAC. Because the counter only increments in one direction (i.e. monotonic) the MAC will always be different since the counter number input to the hash will not be repeated.   Using the ink tank example again, that means that an ink tank cannot be refilled and used again because it cannot be re-authenticated due to the MAC being different because of the incrementing counter.

ATECC508A adds capability to count to 2 Million, and it can use all 15 keys

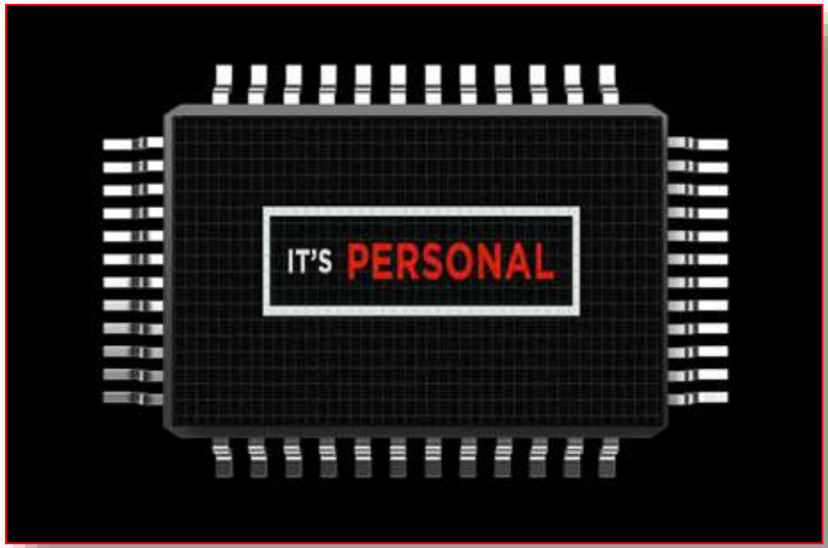
In addition, the counters can be used to perform traditional cryptographic audit functions.



Consumption Tracking

- Attach key usage to monotonic counter
  - When counter is done, key can no longer be used. No back-doors.
- Allows tracking the usage of consumable to prevent re-use/re-load
- Ensures optimal system performance by mandating periodic replacement of the consumable
- AES132 has 16 general purpose 2M count monotonic counters
  - Use separately or attach to keys
- SHA204 & ECC108 have various configurations
  - Single use (paring), 128 uses, Chained to count as high as 800K

58. Secure Personalization



To perform the specific authentication task required by a target application using CryptoAuthentication™ devices, the devices must first be programmed. This programming process is called "personalization" and entails configuring the device with the desired firmware protection profile, which includes secrets such as secret keys. Any way of doing so must protect the keys at every step.

There is a range of ways to personalize the device:

1) Using a socket developed in-house by the customer and issuing commands to every memory location in the device. This is not well suited for volume production

2) Use Atmel Crypto Evaluation Studio (ACES) software tool to issue commands to an AT88CK101 development kit that houses the device in a socket.

3) Use the Atmel AT888CK9000 Secure Personalization Kit. This kit programs up to 5 devices at a time using the ACES tool.

4) Use an approved third-party programmer from System General or Xeltek.

5) Use a secure personalization service from a trained and authorized Atmel partner, such as Avnet, Arrow, and EBV.

6) Use Atmel's "**Secure Personalization Services"**.   Atmel provides the full personalization process that loads the customer's secrets into the volume production devices in a highly secure way. What "secure way" means is that the customer's secrets are never exposed at any point in the process, even to Atmel. Secure algorithms, processes, and hardware security modules are employed to preserve the secrets.

59. Secure Personalization Services

Atmel's **Secure Personalization Service** is designed to transfer the secrets from the customer site to the CryptoAuthentication device without the secrets ever being in the clear, or viewable by party other than the customer. Even Atmel is not able to see the secrets.

Once the configuration of the device has been designed, the customer receives a "Secrets Exchange Package" from Atmel which includes a Personalization Utility program that encrypts the customer's secrets located in an .XML file.   This utility may also perform some formatting and logic testing on the customer's data. There are also a Configuration XML file representing the memory configuration of the device (not including the secrets), and RSA keys that will be used to encrypt the customer's .XML file. These keys are generated on Hardware Security Modules (HSMs) located at the Atmel contractor sites where the final IC package tests are performed. (There can be multiple keys depending on the number of locations where the ICs will be personalized.)

The items enumerated above are used to transfer the customer's secrets between three entities:

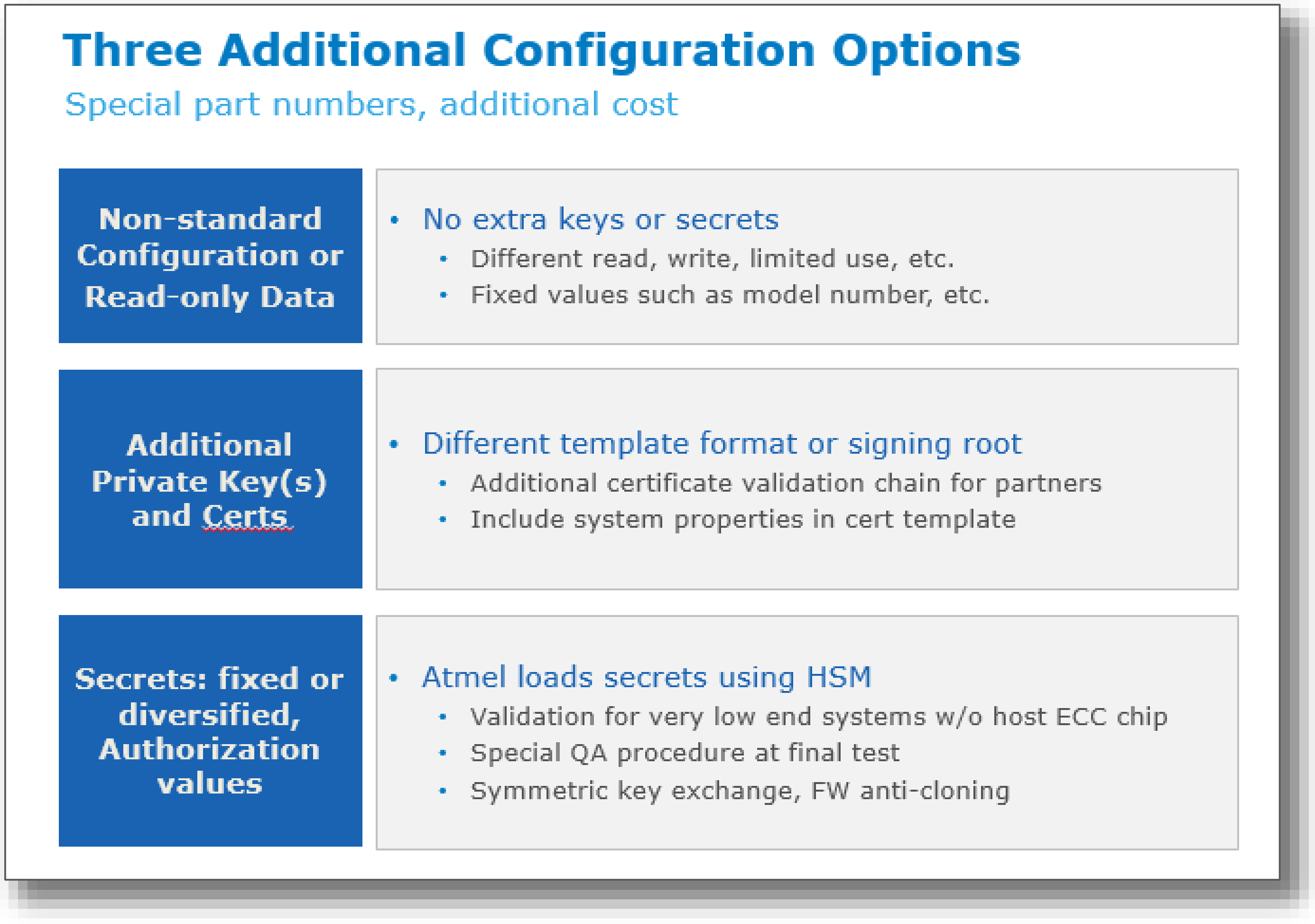
1) The Customer's site

2) The Atmel Hardware Security Module (HSM) at Atmel's production site, and

3) The Atmel CryptoAuthentication™ device itself, on the tester at Atmel's production site.

The concept behind this process is very simple. The customer makes the secret and transfers it to Atmel in **encrypted format** to be stored in the HSM, which is a secure module that not even Atmel can access. When the secrets are to be loaded into the CryptoAuthentication device on the tester, only then are the secrets decrypted inside the **HSM** and immediately re-encrypted to be transferred to the device. Only when they are securely inside the CryptoAuthentication device (which itself is protected hardware (by definition)) are the secrets decrypted and loaded into the assigned memory slots.   The secrets are never in the clear.

60. Additional Configuration Options



Three Additional Configuration Options

Special part numbers, additional cost

Non-standard Configuration or Read-only Data	• No extra keys or secrets • Different read, write, limited use, etc. • Fixed values such as model number, etc.
Additional Private Key(s) and Certs	• Different template format or signing root • Additional certificate validation chain for partners • Include system properties in cert template
Secrets: fixed or diversified, Authorization values	• Atmel loads secrets using HSM • Validation for very low end systems w/o host ECC chip • Special QA procedure at final test • Symmetric key exchange, FW anti-cloning

61. Atmel CryptoAuthentication ™ Devices

The **ATSHA204A** is ideal for cost sensitive applications given its low price. It is perfect for applications that store the same secret in the host and client. It supports the SHA256 cryptography algorithm.

The **ATECC108A/508A** are advantageous for systems with components from different manufacturers because secrets can be easily locked. Also, the devices are optimal for ECDSA and X.509 digital certificate procedures, because those capabilities are built right in. Both are supersets of the ATSHA204A, so they perform all the same symmetric functions. The devices support both SHA256 and ECC. The ATECC508A additionally supports ECDH key agreement making it ideal for network node and IoT applications using encryption methods such as AES.

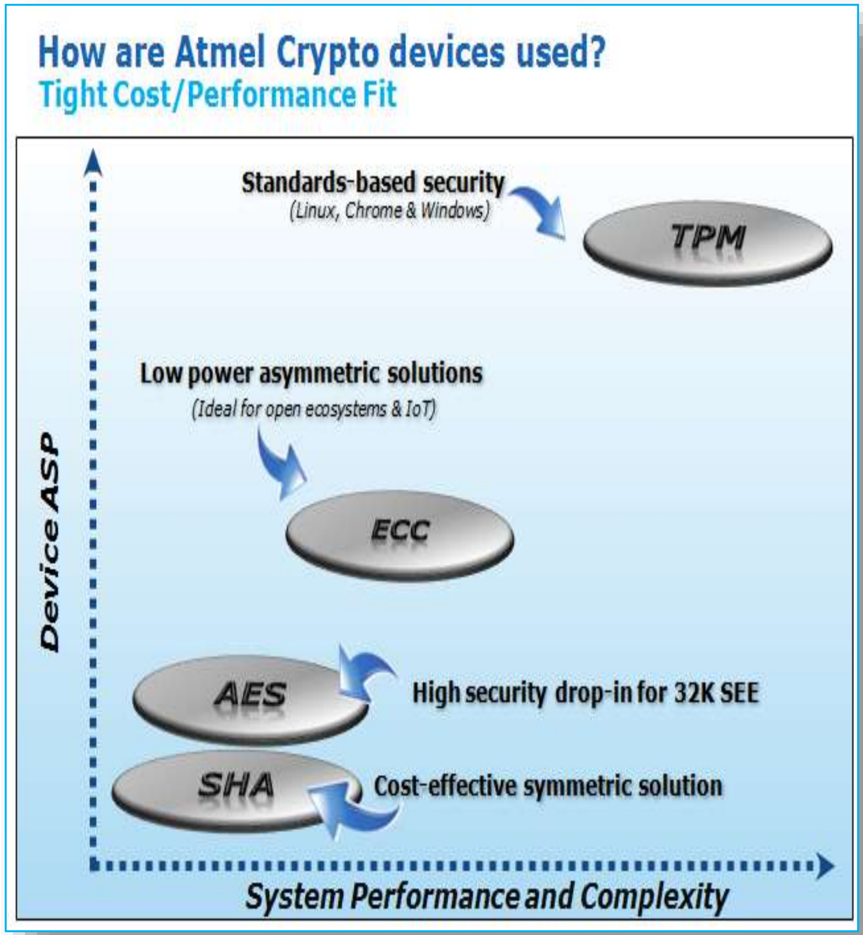
The **ATAES132A** is a secure drop-in for serial 32K EEPROMs. It is a fitting solution for securing up to 4K bytes of data for fingerprints, calibration data, firmware blocks, biometrics and other things. The device supports the AES128 cryptography algorithm.

These are just a few of the high points. There are many other points of differentiation as well, which you can see in more detail in the datasheets

- **Watch Atmel's webinar:** ***Why Investing in Security is Important in Embedded Systems***
- **View Atmel security videos**
- **ATSHA204 datasheet**
- **View More Atmel Security Videos**

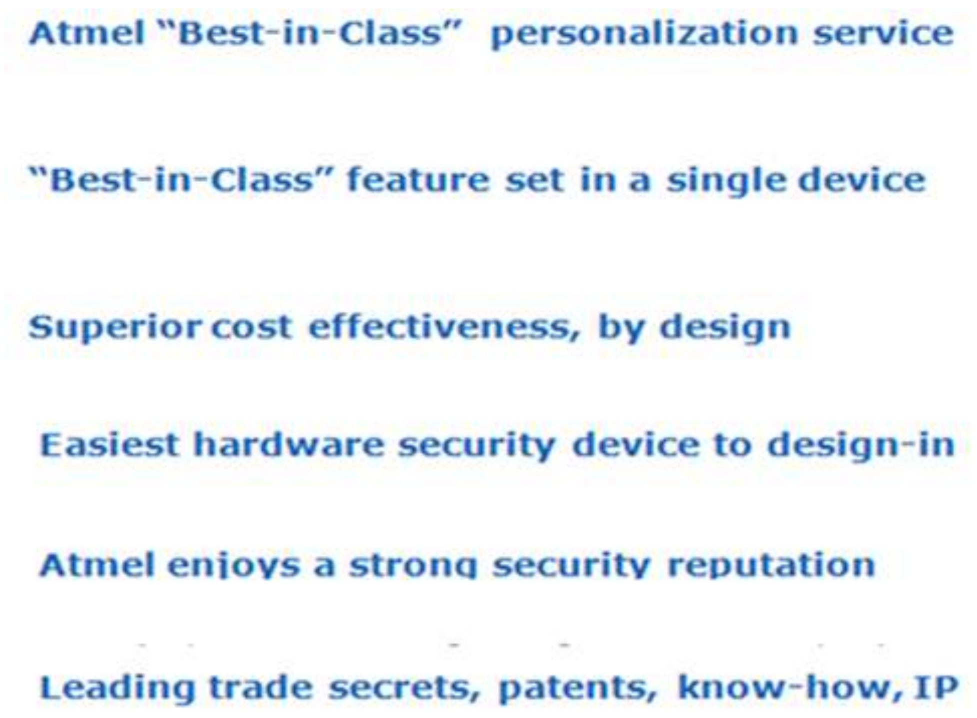
Benefits

- **Price sensitive option (ATSHA204A)**
- **Symmetric options (ATSHA204A & ATECC108A/508A)**
- **Asymmetric authentication (ATECC508A) with ECDSA and ECDH built-in**
- **Asymmetric authentication (ATECC108A) with ECDSA built-in**
- **Add security with a drop in device to standard EEPROM sockets (ATAES132A)**


How are Atmel Crypto devices used?
Tight Cost/Performance Fit
Standards-based security
(Linux, Chrome & Windows)
TPM
Low power asymmetric solutions
(Ideal for open ecosystems & IoT)
ECC
AES
High security drop-in for 32K SEE
SHA
Cost-effective symmetric solution
Device ASP
System Performance and Complexity

62. Competitive Advantages

Lower cost, higher security, better usability



Atmel "Best-in-Class" personalization service

"Best-in-Class" feature set in a single device

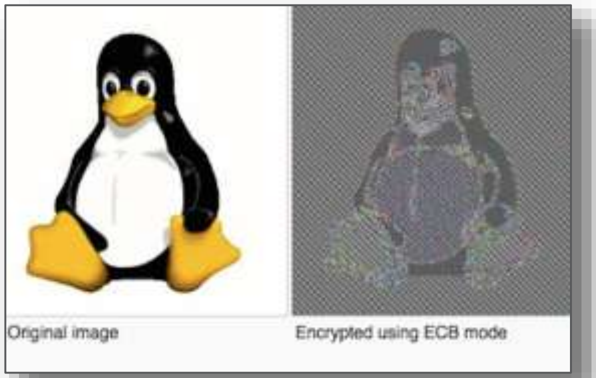
Superior cost effectiveness, by design

Easiest hardware security device to design-in

Atmel enjoys a strong security reputation

Leading trade secrets, patents, know-how, IP

63. AES encryption in the real world



AES is a great algorithm in wide use. It has been carefully studied by legions of cryptanalysts, so it's often assumed that a system which includes AES is secure. **But that assumption isn't always true.** Let's explore three problem areas.

Like all cryptographic systems and algorithms, AES depends on a key. If an attacker can get the key, he or she can impersonate the authentic party, decrypt all the network messages and generally eliminate every aspect of the system security. The key to security is the securing the key. The problem, however, is that very few systems in fact provide a truly secure place to store keys that is truly isolated from attack. (Unless they use protected hardware key storage, of course.)

With the increasingly connected systems, software bugs can easily find keys that you thought you had carefully protected. The Heartbleed bug is a perfect example.

Like all cryptographic algorithms, there are many variations to the way in which AES can be use, which contributed to the fact that a lot of systems have been cracked because an improper mode, protocol or procedure was used. Cryptography must be done correctly to be secure. That is obvious but not always easy. The picture here shows what happens when it is not done right.

The last point is a bit trickier. When encrypting something with AES, most modes require an Initialization Vector (IV). An IV should never be repeated, and in some modes it must be random. There are two problems with a repeated IV: 1) If the attacker could discover the plain text of the first message, he could determine the contents of the second; and 2), if the same message is sent with the same IV, the ciphertext will be the same both times, which is vital information that can be used by attackers to break the encryption.

The big point here is that it is hard to generate a random number. The universe may trend towards entropy but it is filled with patterns along the way, so randomness is not exactly natural. It has to be captured carefully. One famous random number generator used the hash of an image of lava lamps, and for some years an online site ("Lavarand") was supported by Silicon Graphics to provide online numbers.

Assuming you don't have lava lamps and a camera in your system, you might be tempted to use 'random' keystrokes, noise on a signal wire, the current time to the ms, or some similar thing. Problem is, while the resulting numbers appear to be random there are often a limited number of choices. Given how fast modern computers execute, an attacker can try literally millions of possibilities in a few seconds and guess your random number!

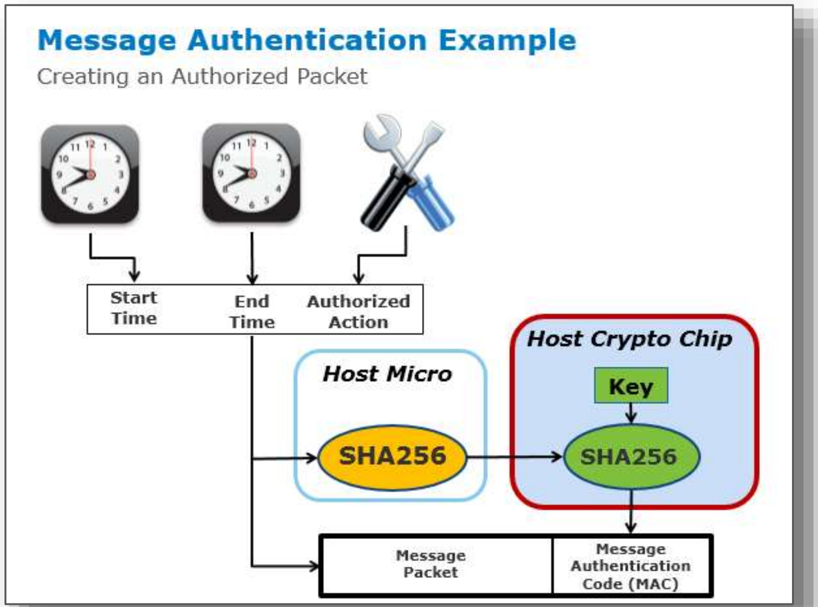
Many designers rely on dedicated hardware cryptographic devices to help resolve this issue. Generally speaking, they offer solutions to the three points mentioned above:

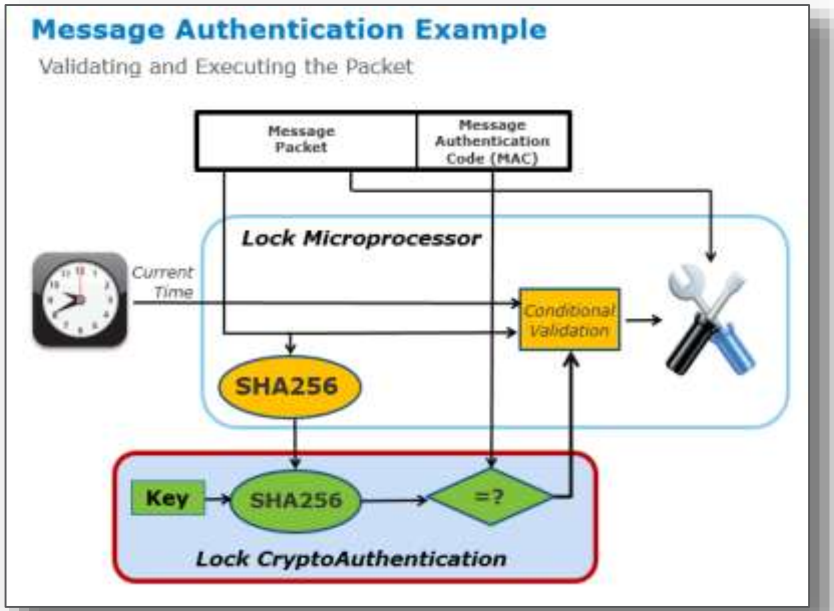
1) **Strong protection for cryptographic keys that is not subject to bugs, malware or other aggressive attacks**
2) **Proper use of modes and protocols for the operations performed within the devices**
3) **High quality random number generators that rely on random physical phenomena and which are rigorously tested**

64. Message Authentication Example

CryptoAuthentication crypto element devices can be used to authenticate a message by creating a simple Message Authentication Code (MAC) and appending that MAC to the message packet. A MAC is a very fundamental cryptographic function. The basic definition of a MAC is the result of a hash (compression) of a key and message. In this example the message is the start time, end, time, and the authorized action. That message is first hashed by the host to put it into the right size to be input to another hash with the key to create the MAC. Now that the MAC is created by the CryptoAuthentication device it is appended to the original (un-hashed) message packet. The packet with the MAC is sent to the other side where it will be validated.

The core of this application is that the execution of the authorized action will depend upon the current time being within the time range specified (i.e. between the start and end times). The original message, which is the action to be authorized, is input to a hash function in the microprocessor just like was done on the other side, to be sized correctly to then be input to the CryptoAuthentication device.


Message Authentication Example
Creating an Authorized Packet
Start Time
End Time
Authorized Action
Host Micro
Host Crypto Chip
Key
SHA256
SHA256
Message Packet
Message Authentication Code (MAC)


Message Authentication Example
Validating and Executing the Packet
Message Packet
Message Authentication Code (MAC)
Lock Microprocessor
Current Time
Conditional Validation
SHA256
Key
SHA256
=?
Lock CryptoAuthentication

The CryptoAuthentication device hashes the hashed (compressed) message with the key stored there to make the MAC. The MAC on that side is compared to the MAC that was sent over from the other side. If the MAC s match, then the messages must be identical, and that proves that they were not tampered with. Now that you know the messages are authentic, the current time can be checked with the range sent over in the original message to see if it is in the specified range. If it is then the authorized action can be executed.

65. *Intrusion Detection Feature* *(ATECC108A & ATECC508A)*

The ATECC108A and ATECC508A devices have the ability to disable authentication of chosen keys if an intrusion is detected. Automatic disabling of authorization keys in the case of a physical intrusion helps businesses and consumers avoid or mitigate losses.

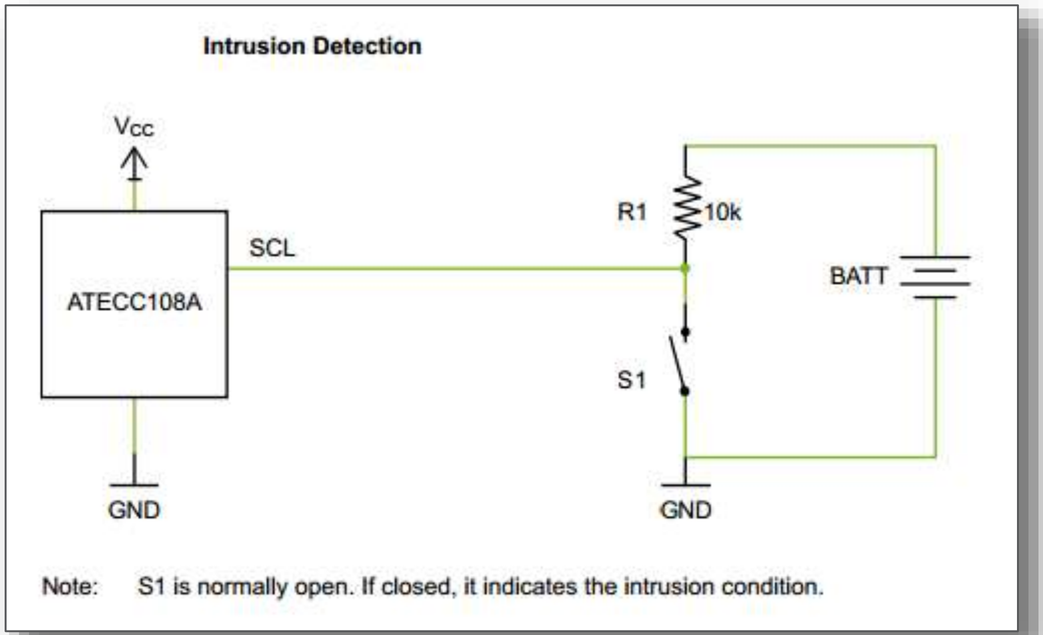
When the ATECC108A or ATECC508A is used in the Intrusion mode, it immediately and automatically stops unauthorized access to the keys in the device. Intrusions can take the form of opening or closing a door without prior authorization, proximity alerts, automobile break-ins, or even dead-man switches in the case of unauthorized machine operations.

If the use of a system feature is tied to a key in the ATECC108A device, by the automatic disabling a key when an intrusion is detected, the use of the key and associated feature will automatically be disabled as well since it can no longer be authorized. The intrusion detection solution is deployed in two parts:

1. **Perform a one-time configuration of the intrusion detect mode on the ATECC108A, and define which keys should be disabled when an intrusion is detected.**

2. **Arm the device to detect intrusion and disable chosen keys**.

Note: The intrusion detection is only available for the versions of the devices supporting Single-Wire Interface ("SWI") communication since it co-opts the SCL pin used in 2-wire communication. Devices configured for SWI, the SCL pin can be configured as a GPIO pin (an input in the case of intrusion trigger).


Intrusion Detection
Vcc
ATECC108A
SCL
R1
10k
BATT
S1
GND
GND
Note: S1 is normally open. If closed, it indicates the intrusion condition.

66. *Feature Summary* of ATMEL ECC CryptoAuthentication devices (ATECC108A and ATECC508A)

All of these are optional

Host Mode

- Same device can both generate and verify P256 ECDSA signature
- Verify takes ~25ms with 500MHz Cortex A5, <50ms ATECC108

Store public key validation

- Additional NV per public keys, stores key validation status
- Subsequent authentication to same node requires just a single sign/verify
- Can support X.509 format signatures via on-board SHA command

Fully Compatible with SHA256-based ATSHA204

- Full suite of symmetric authentication/validation

SHA256 hash command

- ~9ms per 64 byte block

Compressed certificates

- Store information necessary to re-create full x.509 certificate in only 72 bytes
- Can store 8 on-chip, others off-chip
- Use standard certificate validation software
- Single (Limited) Use Keys, (if so configured)
- Once keys have been used the desired number of times, they are no longer usable

Individual slot locking

- Once locked, a slot can no longer be changed. Store model information, network address, etc.

Random nonce requirement

- Prevents replay attacks by requiring that the cryptographic transaction includes a random number
- Key authorization
- Requires knowledge of appropriate key to enable use of target key on a per-use basis. The authorization status is not stored in NVM.
- Either symmetric or asymmetric validation
- One use model is password authorization

67. *Feature Summary* of ATMEL ECC CryptoAuthentication devices (ATECC108A and ATECC508A)

All of these are optional

Tamper (intrusion) input pin

- Connect to switches on case, metal shields over device, etc. to detect physical 'attack' on the system
- Typically useful where battery power is always present

Authenticated output

- Driven to programmable default level on power-up, then to opposite state after validation (symmetric or asymmetric)
- Enable portions of system, status indication, etc.

GPIO

- 2.0 x 3.0 x 0.6mm, 8-lead UDFN is preferred
- Totem-pole output capable of driving dual-color LED. Retains state during sleep

Selectable input thresholds

- Useful if chip supply is higher than that of micro (e.g. 1.8V)

CRC-16 protected IO

- Results can be re-transmitted without re-computation (or ignored)
- CRC-16 source code software in library on website

Written by Atmel during manufacture

- Cannot be changed in the field

24 bit Fixed for customer Devices

- Included in all symmetric calculations and some asymmetric calcs
- Sold only to customer or authorized third parties

8 bit unique ID

- Guaranteed to be unique for all CryptoAuthentication devices

68. *Atmel CryptoAuthentication ™ Devices:* ***ATECC108A***

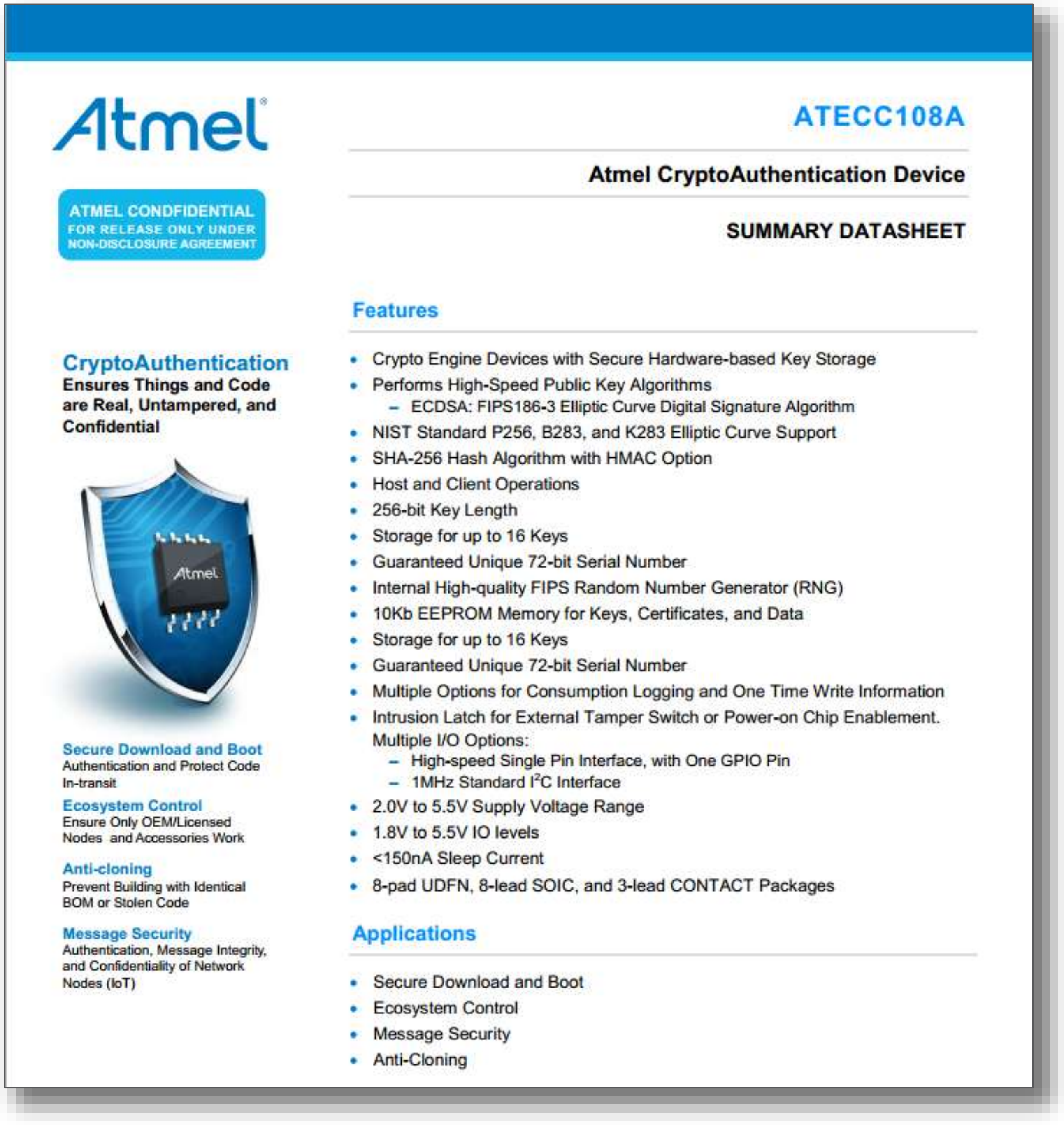
The ATECC108A supports full 256-bit Elliptic Curve Cryptography. A key feature is there is not any need for secure storage in the host. The ATECC108A includes an EEPROM array for storage of up to 16 keys, miscellaneous read/write, read-only or secret data, consumption logging, and security configurations. Access to memory can be restricted in a variety of ways and then the configuration can be locked to prevent changes.

The ATECC108A features defense mechanisms to prevent physical attacks on the device or logical attacks on the data transmitted between the device and the system. Hardware restrictions on the keys generation or use provide further defenses. Access to the device is made through a standard I2C Interface at speeds of up to 1Mb/. It also supports a Single-Wire Interface (SWI), which can reduce the number of GPIOs required on the system MCU. Multiple ATECC108A devices can share the same bus, which saves processor GPIO usage in systems with multiple clients.

Using the ATECC108A's cryptographic protocols, a host system or remote server can verify a signature to prove that the serial number is authentic. The ATECC108A can generate high-quality FIPS random numbers for any purpose including ensuring that that replay attacks (i.e. re-transmitting a previously successful transaction) always fail. System integration is easy due to a wide supply voltage range (of 2.0V – 5.5V) and an ultra-low sleep current (of <150nA).

Benefits

- **Easy way to run Elliptic Curve Digital Signature Algorithm (ECDSA) Sign-Verify operations**
- **Authentication without the need for secure storage in the host**
- **No requirement for high speed computing in client devices**



Atmel

ATMEL CONDFIDENTIAL
FOR RELEASE ONLY UNDER
NON-DISCLOSURE AGREEMENT

ATECC108A

Atmel CryptoAuthentication Device

SUMMARY DATASHEET

CryptoAuthentication
Ensures Things and Code are Real, Untampered, and Confidential

Secure Download and Boot
Authentication and Protect Code In-transit

Ecosystem Control
Ensure Only OEM/Licensed Nodes and Accessories Work

Anti-cloning
Prevent Building with Identical BOM or Stolen Code

Message Security
Authentication, Message Integrity, and Confidentiality of Network Nodes (IoT)

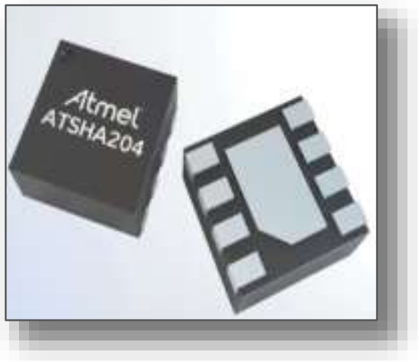
Features

- Crypto Engine Devices with Secure Hardware-based Key Storage
- Performs High-Speed Public Key Algorithms
  - ECDSA: FIPS186-3 Elliptic Curve Digital Signature Algorithm
- NIST Standard P256, B283, and K283 Elliptic Curve Support
- SHA-256 Hash Algorithm with HMAC Option
- Host and Client Operations
- 256-bit Key Length
- Storage for up to 16 Keys
- Guaranteed Unique 72-bit Serial Number
- Internal High-quality FIPS Random Number Generator (RNG)
- 10Kb EEPROM Memory for Keys, Certificates, and Data
- Storage for up to 16 Keys
- Guaranteed Unique 72-bit Serial Number
- Multiple Options for Consumption Logging and One Time Write Information
- Intrusion Latch for External Tamper Switch or Power-on Chip Enablement. Multiple I/O Options:
  - High-speed Single Pin Interface, with One GPIO Pin
  - 1MHz Standard I^{2}C Interface
- 2.0V to 5.5V Supply Voltage Range
- 1.8V to 5.5V IO levels
- <150nA Sleep Current
- 8-pad UDFN, 8-lead SOIC, and 3-lead CONTACT Packages

Applications

- Secure Download and Boot
- Ecosystem Control
- Message Security
- Anti-Cloning

69. Atmel CryptoAuthentication ™ Devices: ATSHA204A



The ATSHA204 has been architected to provide flexible user-configured security to enable a wide range of authentication models, and is easy and timely to design in with no crypto expertise required. The ATSHA204A supports the SHA256 standard and is the most cost effective solution in the portfolio today. The first CryptoAuthentication product to integrate the SHA-256 hash algorithm with a 4.5Kb EEPROM, the ATSHA204A provides robust hardware authentication and secure key/data storage. Features such as small outline plastic package and a single-wire interface make the Atmel ATSHA204 ideal for handheld electronic systems or any space-constrained embedded system.

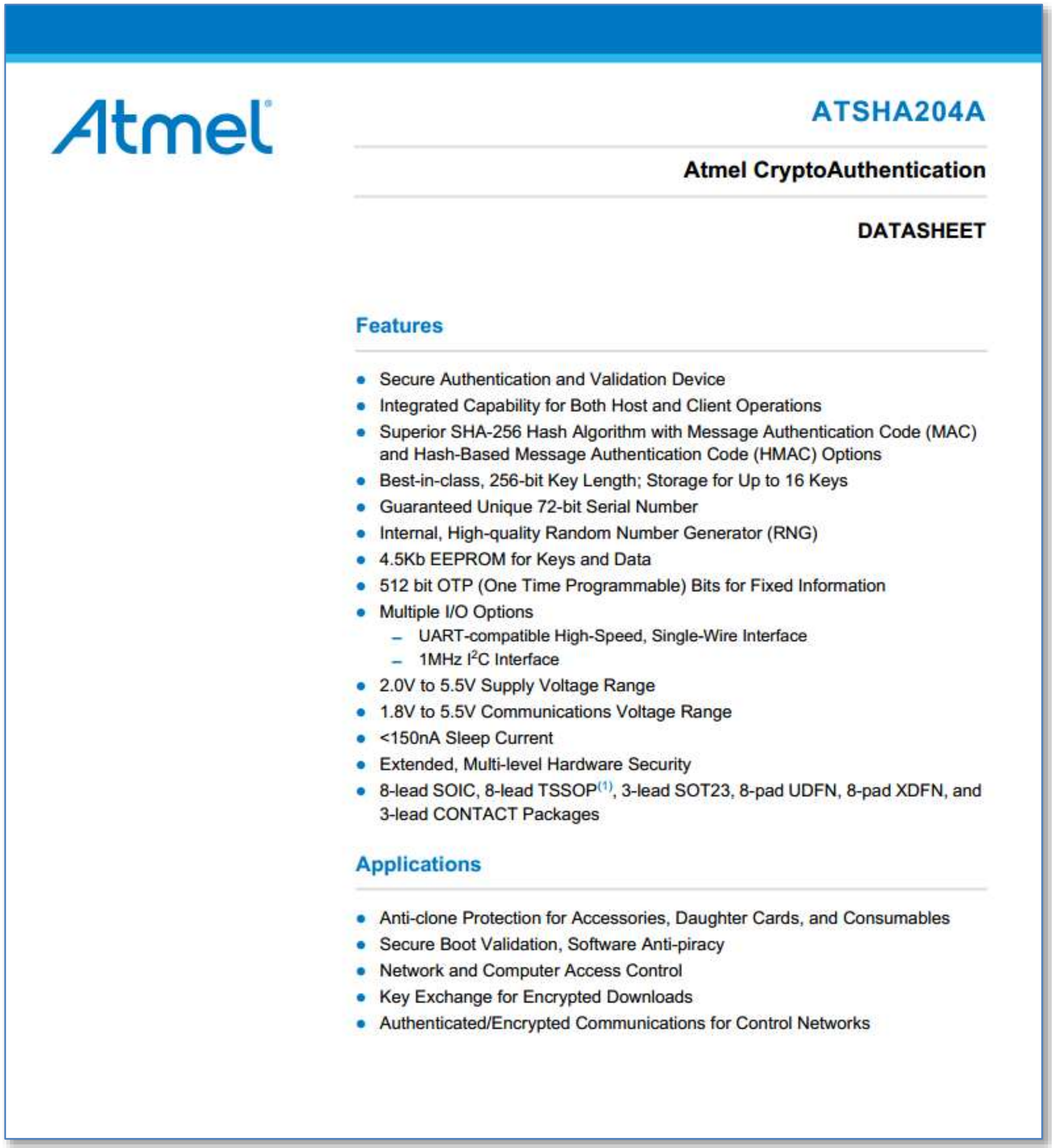
Implementing host-side security to provide a full system solution is now easier than ever. The devices include a client and host security capability that offloads key storage and the execution algorithms from the microcontroller, significantly reducing both system cost and complexity. CryptoAuthentication incorporates the latest security features developed from a long history of security IC design expertise. The devices have full metal shields over the entire internal circuitry, so that if an attacker cuts or shorts any trace in the shield, the device stops functioning. Additional security features include internal clocks and voltage generation, encrypted memories, tamper detection and fully secure production test methods.

Features include:

- Superior SHA-256 NIST-approved algorithm
- Secure authentication and key exchange
- Integrated capability for Host and Client
- Superior SHA-256 Hash algorithm with Message Authentication Code (MAC) and Hash-Based Message Authentication Code (HMAC) options
- Best-in-class, 256-bit key length; storage for up to 16 keys
- Guaranteed unique 72-bit serial number
- 4.5Kb EEPROM for keys and data
- 512 bit OTP (One Time Programmable) bits for fixed information
- 2.0V to 5.5V supply, and 1.8V to 5.5V communications voltage ranges, <150nA sleep current
- 8-SOIC, 8- TSSOP, 3- SOT23, 8-UDFN, and 3-lead Contact packages
- Single-wire and I^{2}C interface
- Secure personalization

Benefits

- **Cost effective symmetric authentication solution**
- **Fast authentication**
- **No need to become a crypto expert**
- **Easy key management**



Atmel

ATSHA204A

Atmel CryptoAuthentication

DATASHEET

Features

- Secure Authentication and Validation Device
- Integrated Capability for Both Host and Client Operations
- Superior SHA-256 Hash Algorithm with Message Authentication Code (MAC) and Hash-Based Message Authentication Code (HMAC) Options
- Best-in-class, 256-bit Key Length; Storage for Up to 16 Keys
- Guaranteed Unique 72-bit Serial Number
- Internal, High-quality Random Number Generator (RNG)
- 4.5Kb EEPROM for Keys and Data
- 512 bit OTP (One Time Programmable) Bits for Fixed Information
- Multiple I/O Options
  - UART-compatible High-Speed, Single-Wire Interface
  - 1MHz I^{2}C Interface
- 2.0V to 5.5V Supply Voltage Range
- 1.8V to 5.5V Communications Voltage Range
- <150nA Sleep Current
- Extended, Multi-level Hardware Security
- 8-lead SOIC, 8-lead TSSOP(1), 3-lead SOT23, 8-pad UDFN, 8-pad XDFN, and 3-lead CONTACT Packages

Applications

- Anti-clone Protection for Accessories, Daughter Cards, and Consumables
- Secure Boot Validation, Software Anti-piracy
- Network and Computer Access Control
- Key Exchange for Encrypted Downloads
- Authenticated/Encrypted Communications for Control Networks

70. Atmel CryptoAuthentication ™ Devices: ATAES132A

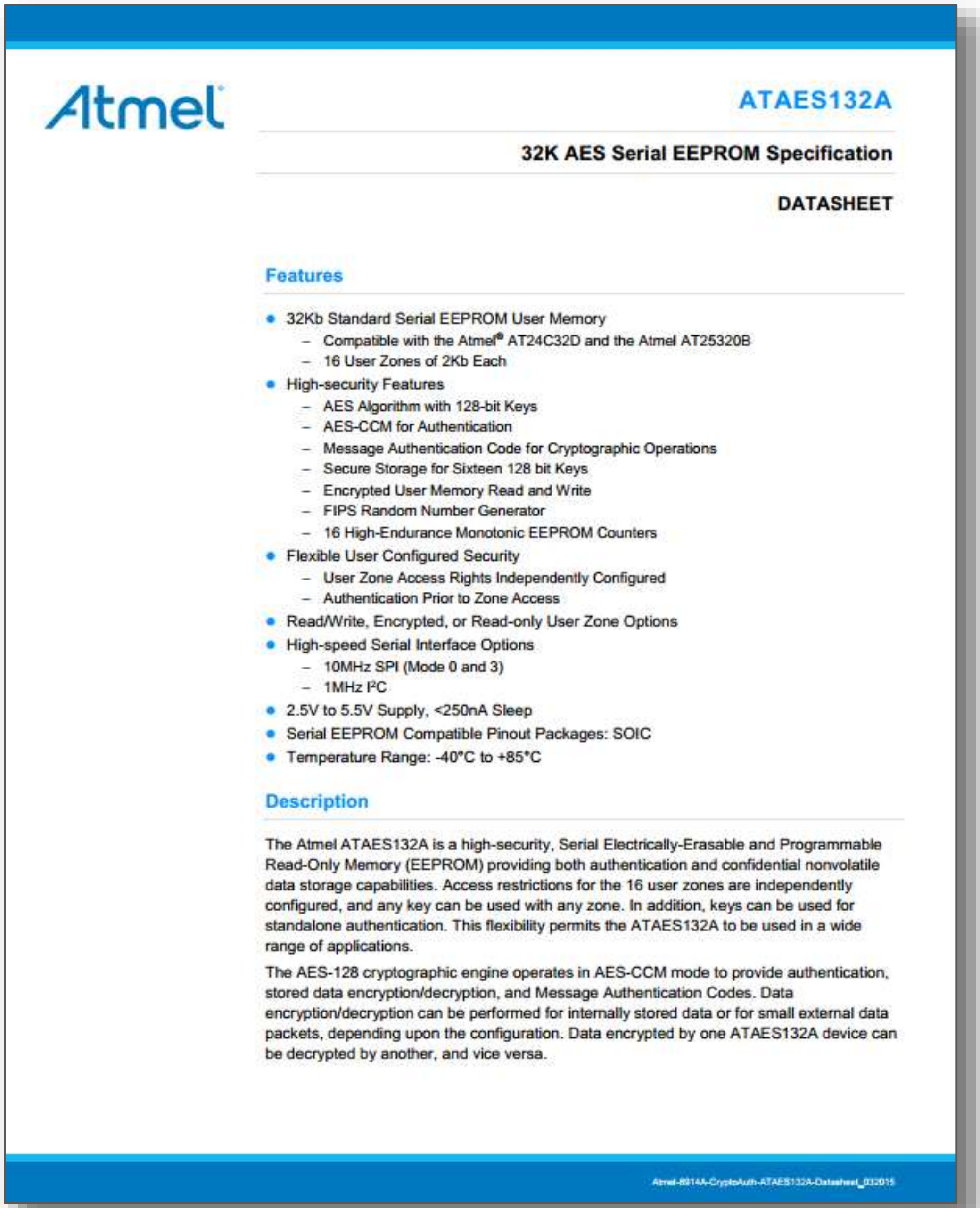
The ATAES132A is a very fast, high-security, serial 32 K EEPROM that enables authentication and confidential nonvolatile data storage. It is a direct drop-in for industry standard serial EEPROMS, and supports the AES cryptography standard. Access restrictions for the 16 user zones are independently configured and any key can be used with any zone. Keys can also be used for standalone authentication. Such flexibility permits the ATAES132 to be used in a wide range of applications. The AES-128 cryptographic engine operates in AES-CCM mode to provide authentication, stored data encryption/decryption, and Message Authentication Codes (MACs). Data encryption/decryption can be performed for internally stored data or for small external data packets, depending upon the configuration. Data encrypted by one ATAES132 device can be decrypted by another, and vice versa. Extended security functions are accessed by sending command packets to the ATAES132 using standard Write instructions and reading responses using standard Read instructions. The device incorporates multiple physical security mechanisms to prevent release of the internally stored secrets. Secure personalization facilitates 3rd-party manufacturing. The device provides sixteen high-endurance monotonic EEPROM Counters. The Configuration Memory registers control access to the User Memory, as well as the restrictions on Key and Counter functionality. The User Memory can be accessed directly with standard SPI or I2C commands. The device's Secure Serial EEPROM architecture and packages compatible with standard SPI and I2C EEPROM footprints allow insertion into many existing Serial EEPROM applications.

Features include:

- 32Kb Standard Serial EEPROM User Memory (16 User Zones of 2Kb)
- AES Algorithm with 128-bit Keys; AES-CCM for Authentication
- MAC for Cryptographic Operations
- Secure Storage for 16 128 bit Keys
- Encrypted User Memory Read and Write
- FIPS Random Number Generator (RNG)
- 16 High-Endurance Monotonic EEPROM Counters
- User Zone Access Rights Independently Configured
- Authentication Prior to Zone Access
- Read/Write, Encrypted, or Read-only User Zone Options
- SPI and I2C Interface Options
- 2.5V to 5.5V Supply, <250nA Sleep
- Serial EEPROM Compatible Pinout (SOIC, SOP, or UDFN)

Benefits

- **Drops into existing sockets to upgrade for high security**
- **Authentication plus encryption in a single protocol**



Atmel

ATAES132A

32K AES Serial EEPROM Specification

DATASHEET

Features

- 32Kb Standard Serial EEPROM User Memory
  - Compatible with the Atmel® AT24C32D and the Atmel AT25320B
  - 16 User Zones of 2Kb Each
- High-security Features
  - AES Algorithm with 128-bit Keys
  - AES-CCM for Authentication
  - Message Authentication Code for Cryptographic Operations
  - Secure Storage for Sixteen 128 bit Keys
  - Encrypted User Memory Read and Write
  - FIPS Random Number Generator
  - 16 High-Endurance Monotonic EEPROM Counters
- Flexible User Configured Security
  - User Zone Access Rights Independently Configured
  - Authentication Prior to Zone Access
- Read/Write, Encrypted, or Read-only User Zone Options
- High-speed Serial Interface Options
  - 10MHz SPI (Mode 0 and 3)
  - 1MHz I²C
- 2.5V to 5.5V Supply, <250nA Sleep
- Serial EEPROM Compatible Pinout Packages: SOIC
- Temperature Range: -40°C to +85°C

Description

The Atmel ATAES132A is a high-security, Serial Electrically-Erasable and Programmable Read-Only Memory (EEPROM) providing both authentication and confidential nonvolatile data storage capabilities. Access restrictions for the 16 user zones are independently configured, and any key can be used with any zone. In addition, keys can be used for standalone authentication. This flexibility permits the ATAES132A to be used in a wide range of applications.

The AES-128 cryptographic engine operates in AES-CCM mode to provide authentication, stored data encryption/decryption, and Message Authentication Codes. Data encryption/decryption can be performed for internally stored data or for small external data packets, depending upon the configuration. Data encrypted by one ATAES132A device can be decrypted by another, and vice versa.

Atmel-8914A-CryptoAuth-ATAES132A-Datasheet_032015

71. *Atmel CryptoAuthentication ™ Devices: ATECC508A*

The ATECC508A is the first crypto device to integrate ECDH (Elliptic Curve Diffie–Hellman) key agreement, which makes it easy to add confidentiality (encryption/decryption) to digital systems including Internet of Things (IoT) nodes used in home automation, industrial networking, accessory and consumable authentication, medical, mobile and other applications. In addition to ECDH the ATECC508A also has ECDSA (Elliptic Curve Digital Signature Algorithm) sign-verify capabilities built right in to provide highly secure asymmetric authentication. The combination of ECDH and ECDSA in the ATECC508A makes the device an ideal way to provide all three pillars of security—namely confidentiality, data integrity, and authentication—when used with MCU or MPUs running encryption/ decryption algorithms (such as AES) in software. Similar to all Atmel CryptoAuthentication products, the new ATECC508A employs ultra-secure hardware-based cryptographic key storage and cryptographic countermeasures which are more secure than software-based key storage. This next-generation CryptoAuthentication device is compatible with any microprocessor (MPU) or microcontroller (MCU) including Atmel® | SMART and Atmel AVR® MCUs or MPUs. As with all CryptoAuthentication devices, the ATECCC508A delivers extremely low-power consumption, requires only a single GPIO over a wide voltage range, and has a tiny form factor, making it ideal for a variety of applications including those that require longer battery life and flexible form factors.

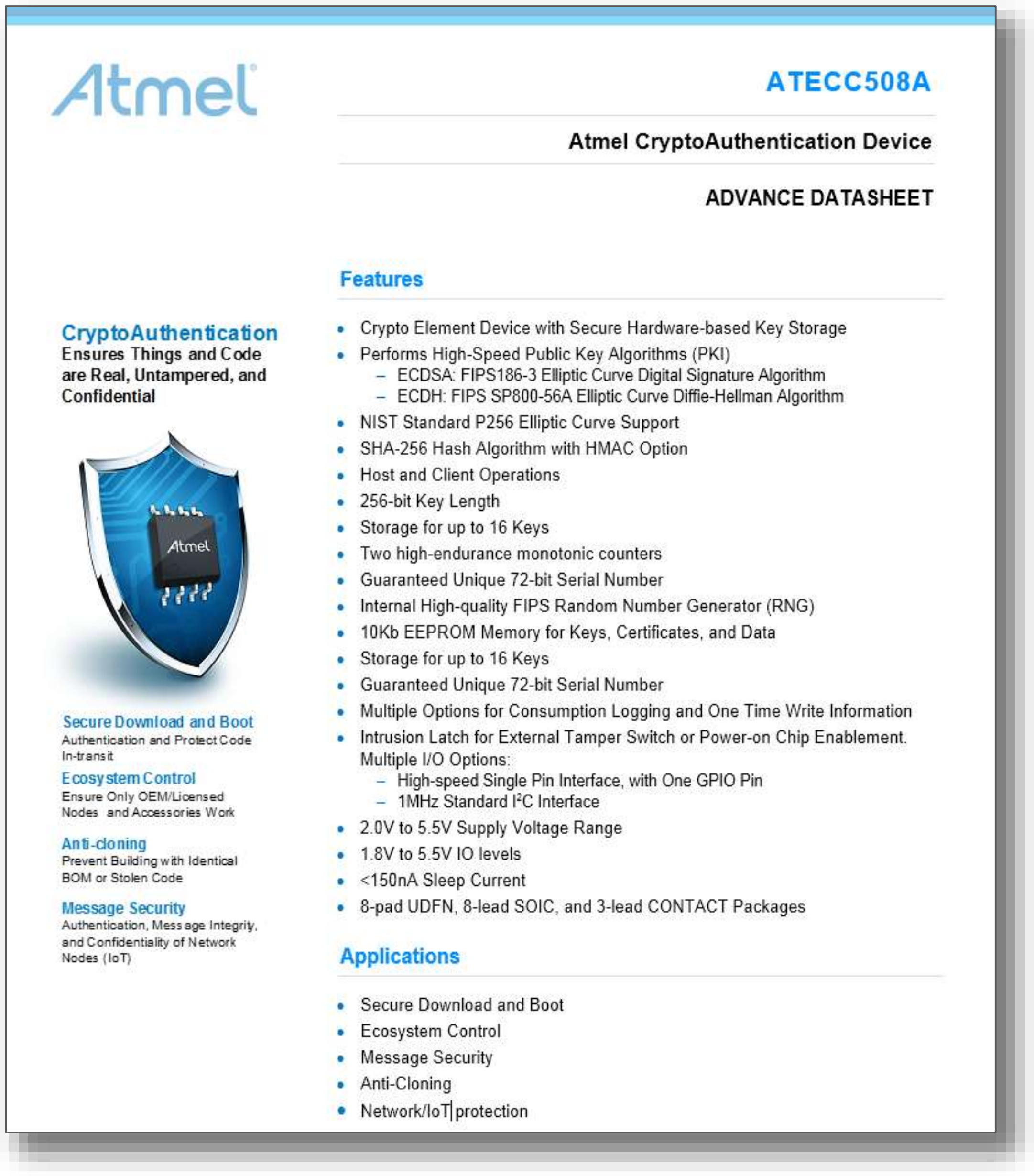
Benefits

- **Easy way to run Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic Curve Diffie-Hellman (ECDH) Key Agreement**
- **Authentication without the need for secure storage in the host**
- **No requirement for high speed computing in client devices**

CryptoAuthentication™ ATECC508A
Crypto Element with ECDH and ECDSA

72. ATECC508A Datasheet

NEW



Atmel

ATECC508A

Atmel CryptoAuthentication Device

ADVANCE DATASHEET

CryptoAuthentication
Ensures Things and Code are Real, Untampered, and Confidential

Secure Download and Boot
Authentication and Protect Code In-transit

Ecosystem Control
Ensure Only OEM/Licensed Nodes and Accessories Work

Anti-cloning
Prevent Building with Identical BOM or Stolen Code

Message Security
Authentication, Message Integrity, and Confidentiality of Network Nodes (IoT)

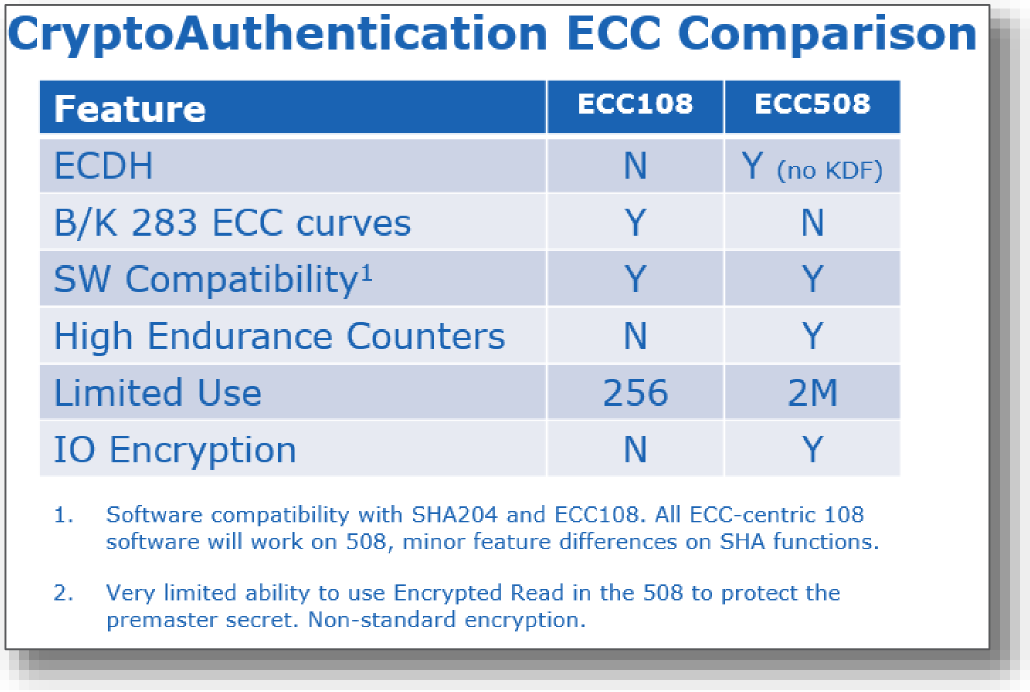
Features

- Crypto Element Device with Secure Hardware-based Key Storage
- Performs High-Speed Public Key Algorithms (PKI)
  - ECDSA: FIPS186-3 Elliptic Curve Digital Signature Algorithm
  - ECDH: FIPS SP800-56A Elliptic Curve Diffie-Hellman Algorithm
- NIST Standard P256 Elliptic Curve Support
- SHA-256 Hash Algorithm with HMAC Option
- Host and Client Operations
- 256-bit Key Length
- Storage for up to 16 Keys
- Two high-endurance monotonic counters
- Guaranteed Unique 72-bit Serial Number
- Internal High-quality FIPS Random Number Generator (RNG)
- 10Kb EEPROM Memory for Keys, Certificates, and Data
- Storage for up to 16 Keys
- Guaranteed Unique 72-bit Serial Number
- Multiple Options for Consumption Logging and One Time Write Information
- Intrusion Latch for External Tamper Switch or Power-on Chip Enablement. Multiple I/O Options:
  - High-speed Single Pin Interface, with One GPIO Pin
  - 1MHz Standard I²C Interface
- 2.0V to 5.5V Supply Voltage Range
- 1.8V to 5.5V IO levels
- <150nA Sleep Current
- 8-pad UDFN, 8-lead SOIC, and 3-lead CONTACT Packages

Applications

- Secure Download and Boot
- Ecosystem Control
- Message Security
- Anti-Cloning
- Network/IoT protection

73. Atmel CryptoAuthentication ™ ECC Device Comparison



CryptoAuthentication ECC Comparison

Feature	ECC108	ECC508
ECDH	N	Y (no KDF)
B/K 283 ECC curves	Y	N
SW Compatibility[1]	Y	Y
High Endurance Counters	N	Y
Limited Use	256	2M
IO Encryption	N	Y

1. Software compatibility with SHA204 and ECC108. All ECC-centric 108 software will work on 508, minor feature differences on SHA functions.
2. Very limited ability to use Encrypted Read in the 508 to protect the premaster secret. Non-standard encryption.

74. CryptoAuthentication Device Selector Guide

NEW

	ATSHA204A	ATECC508A	ATECC108A	ATAES132A
Status	Full Production	Full Production	Full Production	Full Production
Description	Secure authentication and validation device	High speed PKI crypto engine with secure key storage. FIPS186-3 Elliptic Curve Digital Signature Algorithm (ECDSA) for ECC Sign-Verify		High-security, Serial EEPROM providing authentication and confidential nonvolatile data storage.
ECDH		Runs FIPS SP800-56A Elliptic Curve Diffie-Hellman Algorithm		
Function(s)	Authentication	Authentication, and key exchange for confidentiality and data integrity (when used with AES)	Authentication	Encryption / Authentication
Data Sheet Status	Public	NDA		Public
Authentication	SHA , HMAC (Symmetric)	SHA, HMAC, (Symmetric), ECC (Asymmetric)		AES-CCM (Mutual) (Symmetric)
Crypto Algorithms	SHA256	SHA256, ECCP256	SHA256 ,ECC-P256,ECC-B283,ECC-K283	AES128
Key Length	256	SHA=256; ECC=P256	SHA=256; ECC=P256; =K283, =B283	128
Secure boot	SHA	SHA/ECC		-
Key Exchange	SHA	SHA		AES-CCM
Key Authorization	-	SHA, ECC		AES
Key Creation	SHA / Random	SHA / Random		Random
Command Authorization	-	-		AEC-CCM
Encrypted Read/Write	SHA / XOR	SHA / XOR		AES-CCM

75. CryptoAuthentication Device Selector Guide

	ATSHA204A	ATEC508A	ATECC108A	ATAES132A
Commands	13	18		26
Power (max.)	3 mA	16 mA		26 mA
Sleep	< 150nA	< 150nA		<250nA
Vcc	2.0 – 5.5V	2.0 – 5.5V		2.5V to 5.5V
Communications Voltage	1.8V to 5.5V	1.8V to 5.5V		1.8V to 5.5V
EEPROM size	4.5Kb	10 Kb		32K b(User); 2Kb(Keys)
I/O Interface	I2C, Single Wire	High-speed SPI with one GPIO Pin, 1MHz I2C		I2C, SPI
Key Offload & Reload	-	-		Yes
Packages	UDFN8, SOIC8, SOT23-3, 3-Contact (RBH)	UDFN8, SOIC8, 3-Contact (RBH)		UDFN8, SOIC8
Limited Use Keys	9	9		16
Secure Count Max	1K	1K		2M x 16
Authentication Counter	Yes	Yes		Yes
Factory Unique ID	72 bits	72 bits		128 bits

76. CryptoAuthentication Device Selector Guide

	ATSHA204A	ATECC508A / ATECC108A	ATAES132A
Special Features	• Turnkey authentication, validation, key derivation, and password checking • Lowest cost Atmel Encryption / Authentication Device	• Only host has to store public key • Slot locking in field • Programmable random Nonce • 1.75k bit slot for storage • 512 OTP Bits for Fixed Information or Consumption Logging • Intrusion Protection • Full compatibility with ATSHA204	• Full HW & SW compatibility with serial EEPROMs, • 16 counters • Anti-clone circuitry • Mutual authentication w. mutual random Nonce • On-off chip key transfer • On-board encryption/decryption of external data • I2C mode has "Auth Out" pin to show valid authentication
Target Apps	•Cost sensitive apps. •Apps where all components are from same OEM.	• Apps where host hardware changes are difficult. • Apps with many partners or complex ecosystem. • Network Node authentication / IoT	• Apps needing full software compatibility with serial EEPROM or SPI • Apps that need to secure up to 4Kbytes of data for fingerprints, calibration data, firmware blocks, etc.

77. CryptoAuthentication Device Selector Guide

	ATSHA204A	ATECC508A	ATECC108A	ATAES132A
Other Features	Same device for both host and client sides			
	Can be securely personalized			
	Enforces random Nonce			
	Secure test			
	Hardware and PRNG components			
	Internal clock generation			
	Random Number Generator (RNG)			
	Cost effective key-only and secrets-only storage			
	Internally encrypted memories			
	Voltage tamper resistance			
	Common root of trust			
	Active Shield			
	Randomized math operations			
	No probe or test points			
	Data independent crypto execution			
Notes	*1: All devices include additional configuration memory.*			
	2: Monotonic counters can operate independently or be connected to cryptographic keys.			
	3: Specifications are subject to change			

78. Atmel CryptoAuthentication Device Ordering Guide: ATECC108A

Atmel Ordering Code	Package Type	Interface Configuration	Qty per reel /tube (units)	Minimum Order Quantity (MOQ) (units)
ATECC108A-SSHCZ-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	Single-Wire	4,000	12,000
ATECC108A-SSHCZ-B	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC); Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	Single-Wire	100	10,000
ATECC108A-SSHDA-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC); Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^{2}C	4,000	12,000
ATECC108A-SSHDA-B	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^{2}C	100	10,000
ATECC108A-MAHCZ-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) ; Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	Single-Wire	15,000	15,000
ATECC108A-MAHDA-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	I^{2}C	15,000	15,000
ATECC108A-RBHCZ-T	3-lead Contact, Tape and Reel; 2.5mm x 6.5mm body, 2.0mm pitch (Sawn) -40 to 85 degrees C	Single-Wire	5,000	10,000

79. Atmel CryptoAuthentication Device Ordering Guide: ATSHA204A

Atmel Ordering Code	Package Type	Interface Configuration	Qty per reel /tube (units)	Minimum Order Quantity (MOQ) (units)
ATSHA204A-SSHCZ-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	Single-Wire	4,000	12,000
ATSHA204A-SSHDA-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^2C	4,000	12,000
ATSHA204A-SSHDA-B	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^2C	100	10,000
ATSHA204A-XHCZ-T	8-lead TSSOP, Tape and Reel; 8X, 8-lead 4.4mm Body, Plastic Thin Shrink Small Outline Package (TSSOP) -40 to 85 degrees C	Single-Wire	4,000	12,000
ATSHA204A-XHDA-T	8-lead TSSOP, Tape and Reel; 8X, 8-lead 4.4mm Body, Plastic Thin Shrink Small Outline Package (TSSOP) -40 to 85 degrees C	I^2C	4,000	12,000
ATSHA204A-STUCZ-T	3-lead SOT23, Tape and Reel; 3TS1, 3-lead, 1.30mm Body, Plastic Thin Shrink Small Outline Package (Shrink SOT) -40 to 85 degrees C	Single-Wire	15,000	15,000
ATSHA204A-MAHCZ-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) ; Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	Single-Wire	15,000	15,000
ATSHA204A-MAHDA-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) ; Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	I^2C	15,000	15,000
ATSHA204A-RBHCZ-T	3-lead Contact, Tape and Reel; 2.5mm x 6.5mm body, 2.0mm pitch (Sawn) -40 to 85 degrees C	Single-Wire	5000	10000

8 pad UFDN NiPdAu Lead finish. 8-lead SOIC NiPdAu Lead finish

80. Atmel CryptoAuthentication Device Ordering Guide: ATAES132A

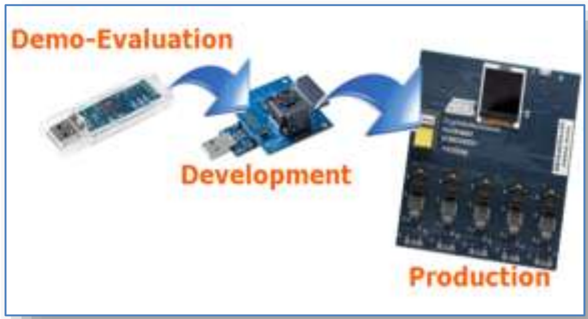
Atmel Ordering Code	Package Type	Interface Configuration	Qty per reel /tube (units)	Minimum Order Quantity (MOQ) (units)
ATAES132A-SH-ER	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^2C	100	10,000
ATAES132A-SH-ER-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^2C	4,000	12,000
ATAES132A-M-ER-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) ; Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	I^2C	15,000	15,000
ATAES132A-SH-EQ	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing ; Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	SPI	100	10,000
ATAES132A-SH-EQ-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gu Wing Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) 40 to 85 degrees C	SPI	4,000	12,000
ATAES132A-MAH-EQ-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) ; Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	SPI	15,000	15,000

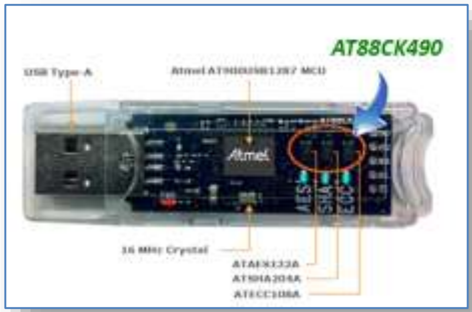
81. Atmel CryptoAuthentication Device Ordering Guide: ATECC508A

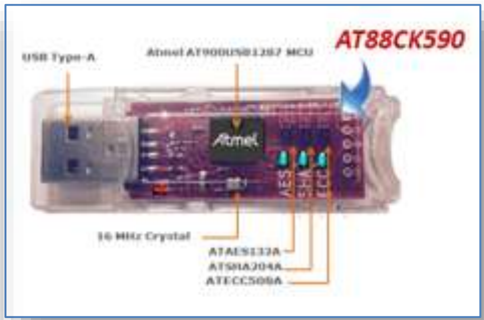
Atmel Ordering Code	Package Type	Interface Configuration	Qty per reel /tube (units	Minimum Order Quantity (MOQ) (units)
ATECC508A-SSHCZ-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	Single-Wire	4,000	12,000
ATECC508A-SSHCZ-B	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC); Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	Single-Wire	100	10,000
ATECC508A-SSHDA-T	8-lead SOIC , Tape and Reel; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC); Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^{2}C	4,000	12,000
ATECC508A-SSHDA-B	8-lead SOIC , Bulk in tubes; 8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC) ; Green, (Pb free, halogen free, ROHS) -40 to 85 degrees C	I^{2}C	100	10,000
ATECC508A-MAHCZ-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat No Lead Package (UDFN) ; Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	Single-Wire	15,000	15,000
ATECC508A-MAHDA-T	8-pad UDFN, Tape and Reel; 8MA2, 8-pad, 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra-Thin Dual Flat, No Lead Package (UDFN) Green (Pb free, halogen free, ROHS), -40 to 85 degrees C	I^{2}C	15,000	15,000
ATECC508A-RBHCZ-T	3-lead Contact, Tape and Reel; 2.5mm x 6.5mm body, 2.0mm pitch (Sawn) -40 to 85 degrees C	Single-Wire	5,000	10,000

82. *Atmel provides tools for every stage*

Atmel provides kits to assist evaluators and developers at every stage from evaluation, to product development, and all the way to small-run production.   All of the Atmel kits run ACES (Atmel CryptoAuthentication Evaluation System) software to configure the CryptoAuthentication ™ devices. The combination of the user friendly hardware and software presents users with exactly what is needed all along the way.







It All Starts with a Demo –Evaluation Kit

The AT88CK490 demo-evaluation kit contains three devices, namely the ATSHA204A, ATECC108A, and ATAES132A (green board). The AT88CK590 kit replaces the ATECC108A with the ATECC508A (red board). The purpose of these boards is to allow the user to use ACES software to configure and evaluate the operation of any of the three crypto devices on the boards. Once the user gets a feel for the operation and performance of the devices, the next tools in the flow are the development kits.

Next is the AT88CK101 or the CryptoAuthXplained PRO Development Kit

The AT88CK101 is a socketed kit thet allows the developer to program CryptoAuthentication devices and then install those devices in their system. Versions that support the various device package types are avialable. Another development kit option is the CryptoAuthXplained Pro, which is a standard 20-pin daughterboard that instantly adds CryptoAuthentication devices to the Atmel Xplained and XpalinedPro development environment boards.

Easily enables small-run production
AT88CK9000
Just press the big yellow button to program 5 devices at the same time.
1
2
3
4
5

AT88CK90000 Secure Personalization Kit (Option)

The Atmel **AT88CK9000** allows programming of small batches of ATSHA204A devices quickly, easily, and cost-effectively. This option decreases the cycle times of prototype, pre-production, and lower-volume production, improving time to market. The AT88CK900 securely programs up to 5 devices at the same time.   Running production is as easy as pressing a button...literally

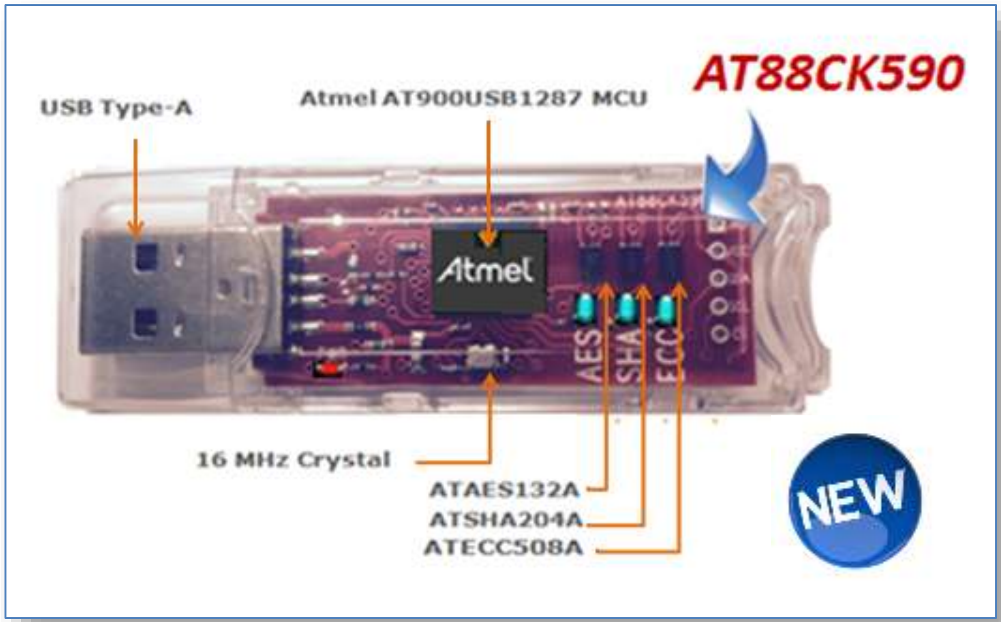
83. AT88CK490 and AT88CK590 Demo-Evaluation kits

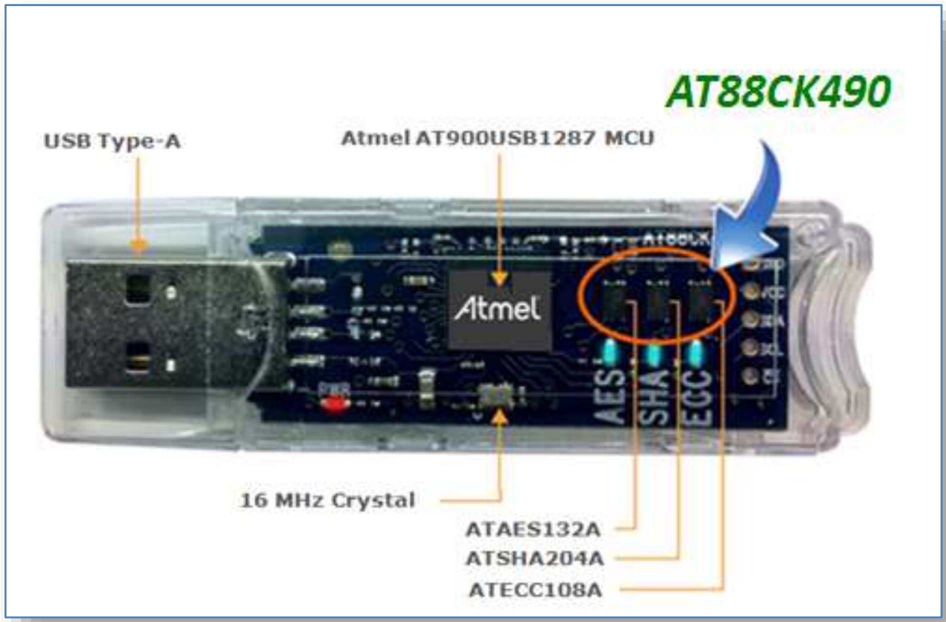
The AT88CK490 demo-evaluation kit contains three devices, namely the ATSHA204A, **ATECC108A,** and ATAES132A (green board). The AT88CK590 kit replaces the ATECC108A with the **ATECC508A (red board)**. The purpose of these boards is to allow the user to use ACES software to configure and evaluate the operation of any of the three crypto devices on the boards.

Once the user gets a feel for the operation and performance of the devices, the next tool in the flow is the AT88CK101 development board

Benefits

- **One dongle demonstrates three parts**
- **Easy to use with ACES configuration environment**
- **Can easily migrate from AT88CK490 and AT88CK590 demo-eval kits to AT88CK101 development kit**





84. CryptoAuthXplained Pro Development Kit

Another development kit Atmel offers is called CryptoAuthXplained Pro, which is a standard 20-pin header (daughterboard) that instantly adds CryptoAuthentication(TM) devices to the Atmel Xplained and XpalinedPro developpment environment boards.

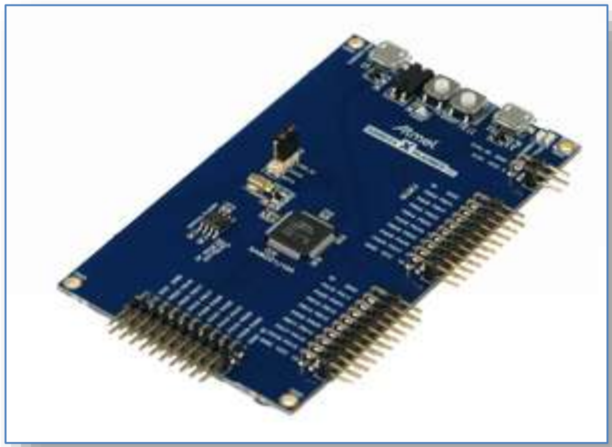
CryptoXplained runs ACES software.

The board supports ATSHA204A, ATECC508A,or ATAES132A in embedded design applications. This kit gives engineers, developers, and decision makers a tool to understand the device architecture and its uses for product authentication, confidential file protection, perform two-factor logons, or prevent software piracy. Complete source code for the Atmel CryptoAuthentication devices is available in ASF (Atmel Software Framework). Schematics, Gerber files, and a bill of materials are also available.

The software that you develop on the PC using this system can also serve as the base for porting code to an embedded microcontroller. The Atmel-provided source code can be easily edited to integrate with ARM or other MCU platforms.

Benefits

- **Works with both Atmel Xplained and Xplained Pro MCU development boards**
- **Users familiar with Xplained and Xplained Pro systems can easily try out CryptoAuthentication devices**
- **Easy to configure devices with ACES configuration environment**



Atmel Xplained Pro Board

85. AT88CK9000 Production Programming Tool

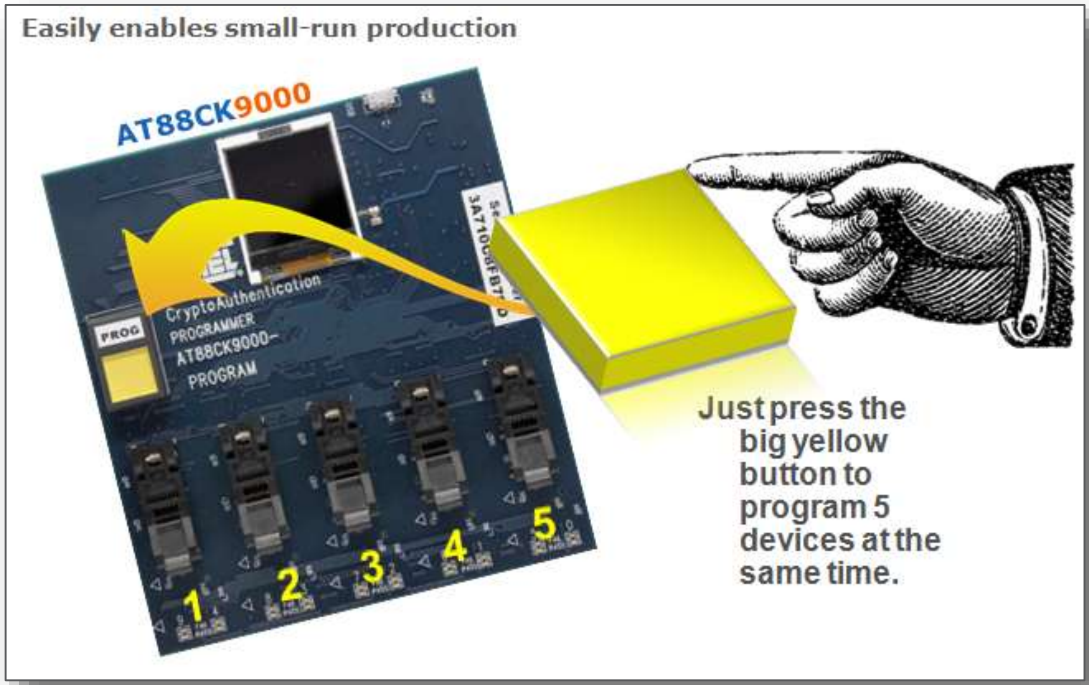
Once the developer has created the desired configurations for the devices, small production runs can be accelerated by use of the AT88CK9000 programming board which securely programs up to 5 devices at the same time. The production configuration image is easily loaded through a USB port. The image is then securely stored on the board as an XML file generated by ACES.

Running production is as easy as pressing a button...literally.

The Atmel **AT88CK9000** Secure Personalization kit for CryptoAuthentication allows programming of small batches of ATSHA204A devices quickly, easily, and cost-effectively. The AT88CK9000 decreases the cycle times of prototype, pre-production, and lower-volume production, improving time to market. The board is powered by the Atmel ARM and AVR® processors and includes a display to provide valuable user feedback. Additionally, a comprehensive User Guide is downloadable from www.atmel.com.

Benefits

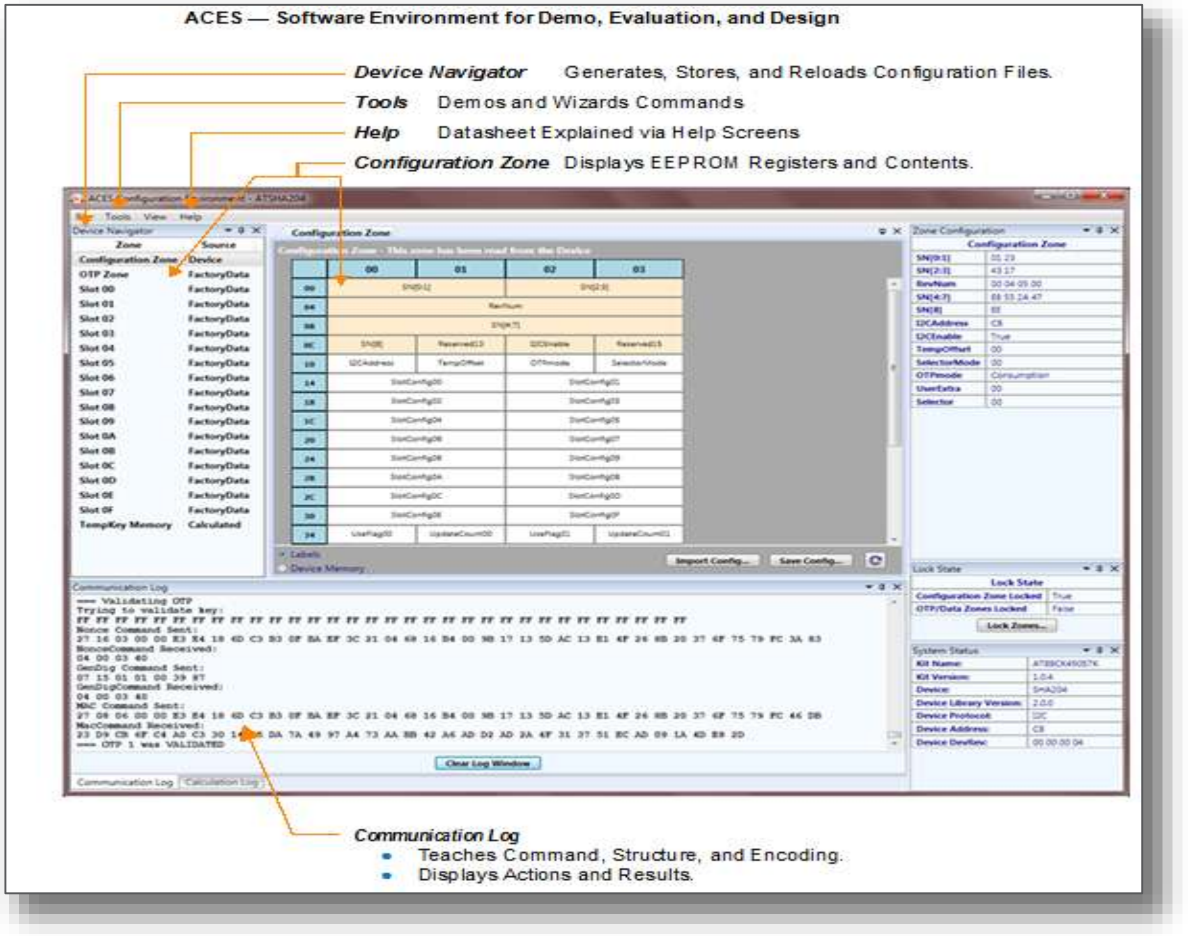
- **Speed time to market by making early production runs**
- **Easy to program devices**
- **Can program up to 5 devices at the same time**


Easily enables small-run production
AT88CK9000
CryptoAuthentication PROGRAMMER AT88CK9000-PROGRAM
PROG
1
2
3
4
5
Just press the big yellow button to program 5 devices at the same time.

Atmel
AT88CK9000 Secure Personalization Kit for CryptoAuthentication™
Universal USB Charger with 1 Port

86. “ACES” (Atmel Crypto Evaluation Studio) Software

http://www.atmel.com/tools/ATMELCRYPTOEVALUATIONSTUDIO_ACES.aspx for detailed information about using ACES. An example of the ACES configuration screen is shown below:


ACES — Software Environment for Demo, Evaluation, and Design
Device Navigator Generates, Stores, and Reloads Configuration Files.
Tools Demos and Wizards Commands
Help Datasheet Explained via Help Screens
Configuration Zone Displays EEPROM Registers and Contents.
Communication Log
Teaches Command, Structure, and Encoding.
Displays Actions and Results.

Benefits

- **ACES works on all Atmel crypto demo, development, and production systems**
- **Wizards and demos make it easy to learn about CryptoAuthentication devices**
- **Demo code can be used as basis of development code**

87. Atmel CryptoAuthentication Kit Ordering Guide

Name	Function	Ordering Code	Detailed Description	Kit Contents	
AT88CK490 Demo-Evaluation Kit	*Very low cost and easy demonstration and evaluation in PC environment*	***AT88CK490***	*Evaluation kit for Atmel ATSHA204A, ATECC108A, and ATAES132A CryptoAuthentication devices.* *User plugs kit into USB port and downloads ACES Demo-Evaluation software*	● ***USB dongle***	
CryptoAuthXplained Pro Daughterboard	*Daughterboard that adds CryptoAuthentication(TM) devices to the Atmel Xplained and XplainedPro series of MCU development boards.*	***ATCryptoAuthXplained Pro***	Standard 20-pin header (daughterboard) with ATSHA204A, ATECC508A, and ATAES132A devices *Runs ACES software*	● ***20-pin daughter-board***	
CryptoAuthXplained Daughterboard	*Daughterboard that adds CryptoAuthentication(TM) devices to the Atmel Xplained and XplainedPro series of MCU development boards.*	***ATCryptoAuthXplained***	Standard 10-pin header (daughterboard) with ATSHA204A, ATECC508A, and ATAES132A devices Runs ACES software	■ ***10-pin daughter-board***	
AT88CK590 Demo-Evaluation Kit	*Very low cost and easy demonstration and evaluation in PC environment*	***AT88CK590***	*Evaluation kit for Atmel ATSHA204A, ATECC508A, and ATAES132A CryptoAuthentication devices.* *User plugs kit into USB port and downloads ACES Demo-Evaluation software*	● ***USB dongle***	

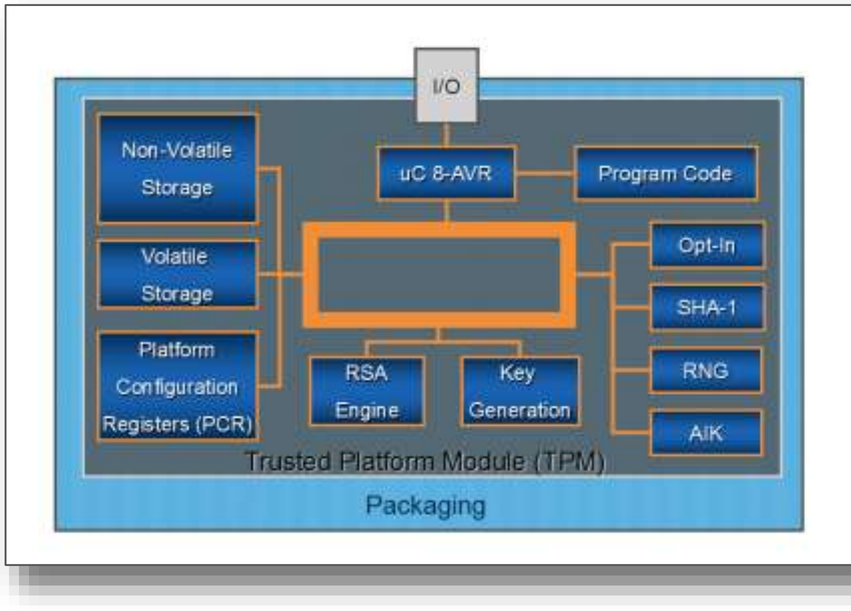
AT88CK101 Development Kit (SOT23)	*Single socket secure development kit. Connects to a PC via USB and also supports Atmel Xplained Pro series*	***AT88CK101SK-TSU-XPRO***	*Socketed kit for software development of security applications on the Atmel ATSHA204A CryptoAuthentication device in SOT-23 packages.* *Runs ACES software*	• *Microbase board** • *Board with 3-lead SOT23 socket* • *IC samples* • *USB extension cable*	
AT88CK101 Development Kit (8-SOIC)	*Single socket secure development kit. Connects to a PC via USB and also supports Atmel Xplained Pro series*	***AT88CK101SK-SSH-XPRO***	*Socketed kit for software development of security applications on the Atmel ATSHA204A, ATECC108A, ATECC508A, and ATAES 132A CryptoAuthentication devices in 8-pin SOIC packages. Runs ACES software*	• *Microbase board** • *Board with 8-lead SOIC socket* • *IC samples* • *USB extension cable*	
AT88CK101 Development Kit (8-UDFN)	*Single socket secure development kit. Connects to a PC via USB and also supports Atmel Xplained Pro series*	***AT88CK101SK-MAH-XPRO***	*Socketed kit for software development of security applications on the Atmel ATSHA204A, ATECC108A, ATECC508A, and ATAES 132A CryptoAuthentication devices in 8-pin UDFN packages. Runs ACES software*	• *Microbase board** • *Board with 8-lead UDFN socket* • *IC samples* • *USB extension cable*	
AT88CK101 Development Kit ***(3-lead RBH)***	*Single socket secure development kit. Connects to a PC via USB and also supports Atmel Xplained Pro series*	***AT88CK101SK-RBH***	*Socketed kit for software development of security applications on the Atmel ATSHA204A, ATECC508A and ATECC108A CryptoAuthentication devices in 3-contact RBH packages. Runs ACES software*	• *Microbase board** • *Board with 3-contact RBH socket* • *IC samples* • *USB extension cable*	

*Note: * The Microbase is an Atmel AVR based AT90USB1287 board that provides a platform to test and develop applications directly with the Atmel CryptoAuthentication IC The board comes with a JTAG port for debug as well as a USB connector to plug into a PC*

Atmel CryptoAuthentication Kit Ordering Guide (Continued)

Name	Function	Ordering Code	Detailed Description	Kit Contents	
AT88CK9000 Production Tool (UDFN8)	Secure personalization production tool for CryptoAuthentication devices in *8-pad UDFN packages*	***AT88CK9000-8MA***	**AT88CK9000** Secure Personalization kit allows programming of small batches of ATSHA204A devices quickly, easily, and cost-effectively. *Runs ACES software*	• *Secure Personalization board* • *Multi-voltage power adapter* • *USB cable*	
AT88CK9000 Production Tool (SOIC8)	Secure personalization production tool for CryptoAuthentication devices in *8-lead SOIC packages*	***AT88CK9000-8SH***	**AT88CK9000** Secure Personalization kit allows programming of small batches of ATSHA204A devices quickly, easily, and cost-effectively. *Runs ACES software*	• *Secure Personalization board* • *Multi-voltage power adapter* • *USB cable*	
AT88CK9000 Production Tool (3-Contact RBH)	Secure personalization production tool for CryptoAuthentication devices in *3-Contact RBH packages*	***AT88CK9000-RBH***	**AT88CK9000** Secure Personalization kit allows programming of small batches of ATSHA204A devices quickly, easily, and cost-effectively. *Runs ACES software*	• *Secure Personalization board* • *Multi-voltage power adapter* • *USB cable*	
AT88CK9000 Production Tool (SOT23-3)	Secure personalization production tool for CryptoAuthentication devices in *3-lead SOT23-3 packages*	***AT88CK9000-TSU***	**AT88CK9000** Secure Personalization kit allows programming of small batches of ATSHA204A devices quickly, easily, and cost-effectively. *Runs ACES software*	• *Secure Personalization board* • *Multi-voltage power adapter* • *USB cable*	

88. *Atmel Trusted Platform Module (TPM)*



The Atmel Trusted Platform Module (TPM) is a complete FIPS 140-2 module certified turnkey solution providing ultra-strong standards-based security for PCs and embedded systems. Primary TPM capabilities include system integrity, authentication, IP protection and secure communication. The core TPM building blocks are the Atmel AVR® microcontroller, Atmel hardware security, and Atmel EEPROM technologies. Security measures also include a variety of circuits to detect and act upon voltage, temperature and frequency tampering. Available in 28-TSSOP and space-saving 4x4mm 32-QFN package options, Atmel TPM devices provide cost effective, standards-based security solutions for all computing devices that are connected to the internet.

Features include:

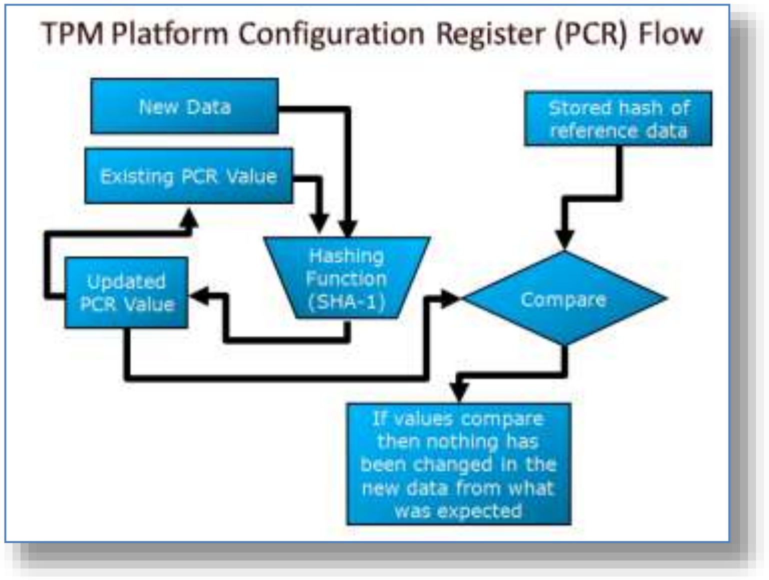
- Full Trusted Computing Group (TCG) v1.2 rev. 116 specification compliance
- 2048-bit RSA hardware crypto accelerator
- Hardware SHA-1 accelerator
- On-chip storage of up to 10 2048-bit RSA key pairs
- High-quality FIPS 140-2 hardware random number generator
- High reliability EEPROM for nonvolatile storage
- Commercial and industrial grade temperature versions
- 3.3 V operation
- BIOS and hardware drivers for Windows and Linux; third-party system and application software.
- Multiple I/O options: LPC, I^{2}C, and SPI

Benefits

- **Turnkey solutions that include nonvolatile storage of keys and secrets.**
- **TCG compliance means superior security and risk management**
- **Hardware security defense mechanisms provide tamper detection and response**
- **Crypto acceleration provides high performance**

89. *Trusted Platform Module (TPM) Security Features*

The TPM is a cryptographic device with heavy cryptographic firepower, such as Platform Configuration Registers, protected user configurable non-volatile storage, an enforced key hierarchy, and the ability to both seal and bind data to a TPM. It does not stop there. Atmel's TPM has a variety of Federal Information Processing Standards (FIPS) 140-2 certified cryptographic algorithms (such as RSA, SHA1, AES, RNG, and HMAC) and various sophisticated physical security counter-measures. The TPM can be used right out of the box with standards-based commands defined by the Trusted Computing Group, along with a set of Atmel-specific commands, which are tested and ready to counter real world attacks.


TPM Platform Configuration Register (PCR) Flow
New Data
Existing PCR Value
Stored hash of reference data
Updated PCR Value
Hashing Function (SHA-1)
Compare
If values compare then nothing has been changed in the new data from what was expected

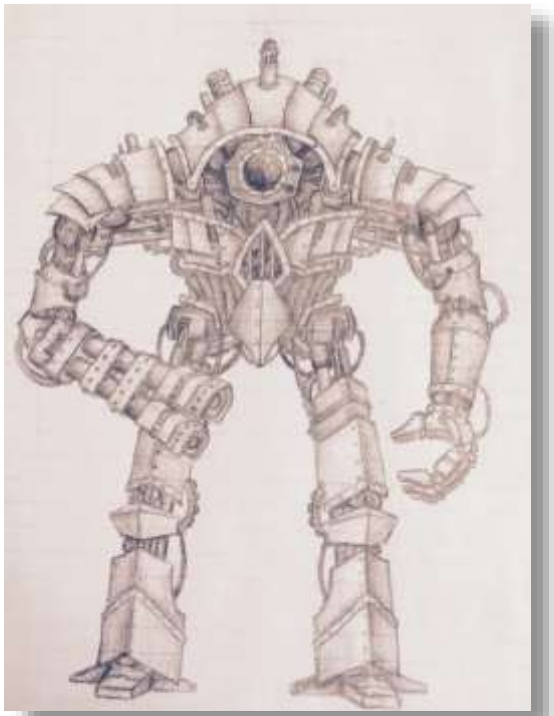
Platform Configuration Registers: One of the important weapons contained in the TPM is a bank of Platform Configuration Registers (PCRs), which use cryptographic hashing functions. These registers can be used to ensure that only trusted code gets loaded at boot time of the system. It does this by using the existing data in a PCR as one input to a hashing function with the other input being new data. The result of that hashing function becomes the new PCR value that will be used as the input to the next hashing function with the next round of new data. This process provides security by continuously changing the value of the PCR. As the PCR value gets updated, the updated values can then be compared with known hash values stored in the system. If the reference values previously stored in the TPM compare correctly with the newly generated PCR values then the inputs to the hashing function (new data in the diagram) are proven to have been exactly the same as the reference inputs whose hash is stored on the TPM. Such matching of the hash values verifies the inputs as being authentic.

Secure Boot: The PCR flow just described is very useful when enforcing secure boot of the system. Unless the hashes match showing that the code is what it supposed to be the code will not be loaded. Even if a byte is added, deleted, or changed; or if a bit is modified then the system will not boot. For secure boot, the data input to the hashing function is a piece of the BIOS (or operating system).

User Configurable Non-Volatile Storage

Another weapon is user-configurable non-volatile storage with multiple configuration options. What this means is that the user is presented with several ways to restrict the access and use of the memory space, such as by password, physical presence of the user, and PCR states. Also the memory space can be set up so that it can be written only once, not read until the next write or startup of the TPM, not written to until the next startup of the TPM, and others.

Trusted Platform Module (TPM) Features (Continued)



Enforced Key Hierarchy. The TPM also incorporates an enforced key hierarchy, which means that keys must have another key acting as a parent key (i.e. a key higher in a hierarchy) for that key to get loaded into the TPM. The authorization information for the parent key needs to be known before the child key can be used, which adds another layer of security.

Binding and Sealing Data. Another part of the TPM's arsenal is the ability to bind and/or seal data to the TPM. A seal operation keeps the data contained (i.e. "sealed") so that it can only be accessed if a particular pre-defined configuration of the system has been reached. This pre-defined configuration is held within the PCRs on the TPM. The TPM will not unseal the data until the platform configuration matches the configuration stored within the PCRs.   A bind operation creates encrypted data blobs (i.e. binary large objects) that are bound to a private key that is held within the TPM. The data within the blob can only be decrypted with the private key in the TPM. Thus, the data is said to be "bound" to that key. Such keys can be reused for different sets of data.

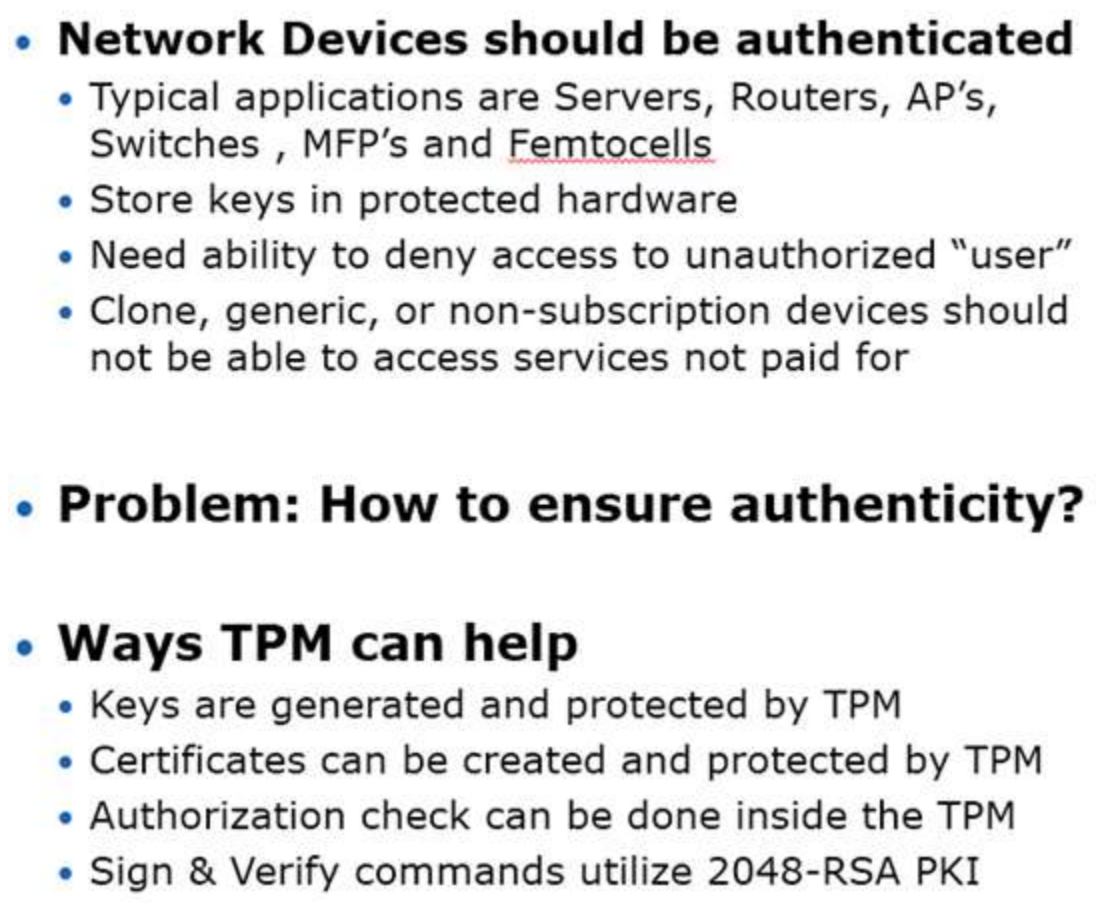
FIPS 140-2 Certified. Atmel has dozens of FIPS 140-2 full module-level certified devices with various I/O's including LPC, SPI, and I^{2}C. The TPM uses a number of FIPS certified algorithms to perform its operations. These standards were developed, tested, and certified by the United States federal government for use in computer systems. The TPM's FIPS certified algorithms include RSA, SHA1, HMAC, AES, RNG and CVL (see the following link: http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm   for more details on Atmel's TPM FIPS certifications).

Active Metal Shield: The TPM has built-in physical armor of its own. A serpentine active metal shield with tamper detection covers the entire device. If someone attempts to penetrate this shield to see the structures beneath it, the TPM can detect this and go into a fault condition that prevents further actions on the TPM.

You might be asking, "Why those functions can't just be done in software?" While some of the protections can be provided in software, software alone is not as robust as a hardware-based system. That is because software has bugs, despite how hard the developers try to eliminate them, and hackers can exploit those bugs to gain access to supposedly secure systems. TPM on the other hand stores secret keys in protected hardware that hackers cannot get access to, and they cannot attack what they cannot see.

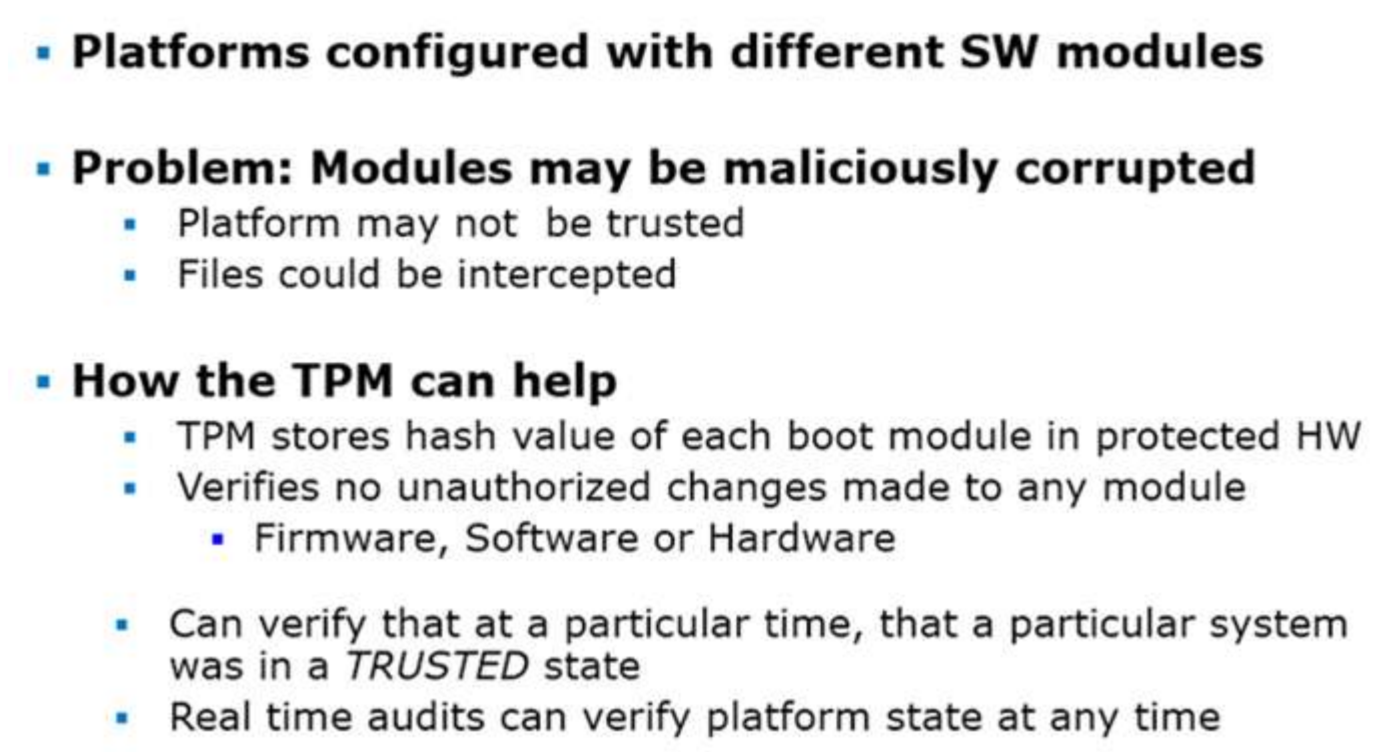
The TPM embeds intelligence via an on-board microcontroller to manage and process cryptographic functions. The commands used by the Atmel TPM have been defined and vetted by the Trusted Computing Group (TCG), which is a global consortium of companies established to define robust standards for hardware security. The Atmel TPM has been successfully tested against TCG's Compliance Test Suite to ensure conformance. Security is also enhanced because secrets never leave the TPM unless they have been encrypted.

90. Authentication Using TPM



- **Network Devices should be authenticated**
  - Typical applications are Servers, Routers, AP's, Switches , MFP's and Femtocells
  - Store keys in protected hardware
  - Need ability to deny access to unauthorized "user"
  - Clone, generic, or non-subscription devices should not be able to access services not paid for
- **Problem: How to ensure authenticity?**
- **Ways TPM can help**
  - Keys are generated and protected by TPM
  - Certificates can be created and protected by TPM
  - Authorization check can be done inside the TPM
  - Sign & Verify commands utilize 2048-RSA PKI

91. Secure Boot Using TPM for Platform Integrity



- **Platforms configured with different SW modules**
- **Problem: Modules may be maliciously corrupted**
  - Platform may not be trusted
  - Files could be intercepted
- **How the TPM can help**
  - TPM stores hash value of each boot module in protected HW
  - Verifies no unauthorized changes made to any module
    - Firmware, Software or Hardware
  - Can verify that at a particular time, that a particular system was in a *TRUSTED* state
  - Real time audits can verify platform state at any time

92. Atmel Trusted Platform Module (TPM) Device Selector Guide

	Trusted Platform Modules	
	AT97SC3204	AT97SC3205
Status	Full Production	
Datasheet Status	Full Data Sheet (NDA required)	
Description	Trusted Platform Module with single-chip strong hardware-based asymmetric-key (RSA) security for PCs and embedded processors. Turnkey system integrating Atmel's AVR® microcontroller, EEPROM, and security technology.	
TCG Version	TCG version 1.2	
Function(s)	Authentication/ Key Generation / Protected key storage	
Authentication Modes	SHA1/ HMAC / RSA	
Crypto Algorithms	RSA 512-2048	
Key Length	2048	
Secure boot	SHA	
Key Exchange	RSA	
Key Authorization		
Key Creation		
Command Authorization		
Commands	103	

93. Atmel Trusted Platform Module (TPM) Device Selector Guide (Continued)

	Trusted Platform Modules	
	AT97SC3204	AT97SC3205
Power (min.)	3 mA	80 uA
Vcc	3.3V	
EEPROM size	2.1Kbytes	
Key Offload & Reload	Yes	
Factory Unique ID	2048 bit Endorsement Key	
Packages	TSSOP28, QFN32	
I/O Interface	I^{2}C (T-version), LPC	I^{2}C (T-version), SPI

94. Atmel Trusted Platform Module (TPM) Device Selector Guide (Continued)

Trusted Platform Modules		
	AT97SC3204	AT97SC3205
Special Features	CC EAL 3+ (4+ Pending)	-
	-	Internal clock generation
	FIPS 140-2 Certified	FIPS 140-2 Certification (Pending)
	FIPS certified Random Number Generator (RNG) available on demand for external use	
	Optional Embedded X.509 certificates (signed endorsement keys)	
	Turnkey with Atmel AVR® microcontroller	
	Common root of trust	
	Active Shield	
	Randomized math ops	
	No probe or test points	
	FIPS 140-2 certified algorithms	
	No need for firmware development (turnkey)	
	Flexible Architecture - multiple security in one chip	
	Data independent crypto execution	
Target Apps	Optimized for standards based Public Key Infrastructure internet / network applications such as Tablets, Phablets, PC's, Access Points, Micro Cells, Routers, Servers, MFP's, x86 based machines, and embedded systems	

95. Atmel Trusted Platform Module (TPM) Device Ordering Guide

Part No.	Pkg Type	Material Description	Interface	Qty per reel /tube	MOQ	Comments
AT97SC3204-X2A1A-10	28 TSSOP, 4.4mm	Lead-free, RoHS v1.2 rev 116, EK, COMM Temp	LPC	1000	1,000	
AT97SC3204-U2A1A-10	28 TSSOP, 4.4mm	Lead-free, RoHS v1.2 rev 116, EK, IND Temp	LPC	1000	1,000	
AT97SC3204-X2A1A-20	28 TSSOP, 4.4mm	Lead-free, RoHS v1.2 rev 116, signed EK, COMM Temp	LPC	1000	1,000	
AT97SC3204-U2A1A-20	28 TSSOP, 4.4mm	Lead-free, RoHS v1.2 rev 116, signed EK, IND Temp	LPC	1000	1,000	
AT97SC3204-X2MA-10	40 QFN	Lead-free, RoHS v1.2 rev 116, EK, COMM Temp	LPC	1000	1,000	
AT97SC3204-U2MA-10	40 QFN	Lead-free, RoHS v1.2 rev 116, EK, IND Temp	LPC	1000	1,000	
AT97SC3204-X2MA-20	40 QFN	Lead-free, RoHS v1.2 rev 116, signed EK, COMM Temp	LPC	1000	1,000	
AT97SC3204-U2MA-20	40 QFN	Lead-free, RoHS v1.2 rev 116, signed EK, IND Temp	LPC	1000	1,000	
AT97SC3204T-X2A1B-10	28 TSSOP	Lead-free, RoHS v1.2 rev 116, EK, COMM Temp	I2C	1000	1,000	Not recommended for new designs. Use AT97SC3205T
AT97SC3204T-U2A1B-10	28 TSSOP	Lead-free, RoHS v1.2 rev 116, with EK, IND Temp	I2C	1000	1,000	Not recommended for new designs. Use AT97SC3205T
AT97SC3204T-X2MB-10	40 QFN	Lead-free, RoHS v1.2 rev 116, with EK, COMM Temp	I2C	1000	1,000	Not recommended for new designs. Use AT97SC3205T
AT97SC3204T-U2MB-10	40 QFN	Lead-free, RoHS v1.2 rev 116, with EK, IND Temp	I2C	1000	1,000	Not recommended for new designs. Use AT97SC3205T

96. Atmel Trusted Platform Module (TPM) Device Ordering Guide

Part No.	Pkg Type	Material Description	Interface	Qty per reel /tube	MOQ
AT97SC3205-X3A12-10	28-pin 4.4mm TSSOP	Commercial Temp (0°C to 70°C) Lead-free, RoHS v1.2 rev 116 SPI TPM	SPI	1000	1,000
AT97SC3205-U3A12-10	28-pin 4.4mm TSSOP	Industrial Temp (-40°C to 85°C) Lead-free, RoHS v1.2 rev 116 SPI TPM	SPI	1000	1,000
AT97SC3205-X3A12-20	28-pin 4.4mm TSSOP	Commercial Temp (0°C to 70°C) Lead-free, RoHS v1.2 rev 116 SPI TPM with Signed Real Mode EK, with 2066B User NV	SPI	1000	1,000
AT97SC3205-U3A12-20	28-pin 4.4mm TSSOP	Industrial Temp (-40°C to 85°C) Lead-free, RoHS v1.2 rev 116 SPI TPM with Signed Real Mode EK, with 2066B User NV	SPI	1000	1,000
AT97SC3205T-X3A13-10	28-pin 4.4mm TSSOP	Commercial Temp (0°C to 70°C) Lead-free, RoHS v1.2 rev 116 I2C TPM with Real Mode EK, with 2066B User NV	I2C	1000	1,000
AT97SC3205T-U3A13-10	28-pin 4.4mm TSSOP	Industrial Temp (-40°C to 85°C) Lead-free, RoHS v1.2 rev 116 I2C TPM with Real Mode EK, with 2066B User NV	I2C	1000	1,000
AT97SC3205T-X3A13-20	28-pin 4.4mm TSSOP	Commercial Temp (0°C to 70°C) Lead-free, RoHS v1.2 rev 116 I2C TPM with signed Real Mode EK, (X.509 cert) with 2066B User NV	I2C	1000	1,000
AT97SC3205T-U3A13-20	28-pin 4.4mm TSSOP	Industrial Temp (-40°C to 85°C) Lead-free, RoHS v1.2 rev 116 I2C TPM with signed Real Mode EK, (X.509 cert) with 2066B User NV	I2C	1000	1,000

97. Trusted Platform Module (TPM) Kits

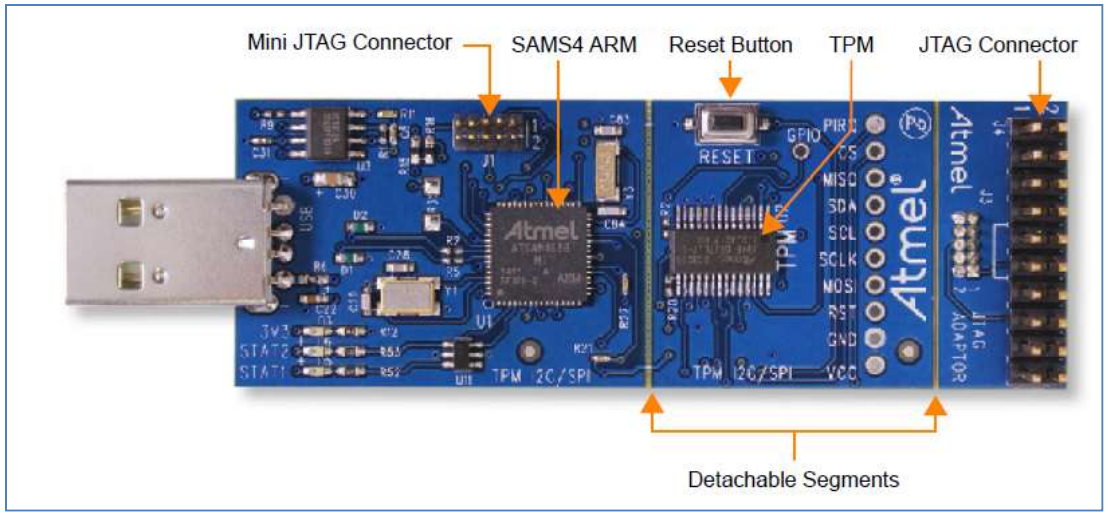
Atmel
Atmel AT97SC3205T-SDK2 I²C
TCG v1.2 Compliant Trusted Platform Module (TPM)
Development Kit
Atmel AT97SC3205T-SDK2 I²C

The Trusted Platform Module Development kits are custom USB based development boards based upon the Atmel SAM4S ARM microcontroller and Atmel **Trusted Platform Module (TPM) devices**. User-friendly demonstration and evaluation software is provided on a USB flash drive. The drive contains demonstration Source Code, hex images for reloading the demo code (if necessary), Trusted Computing Group (TCG) Documentation, and Kit Schematics. A USB extension cable is also included. The kit is updateable with the latest firmware, when available.

The board contains various functional sections including one of the Atmel AT97SC3204 TPM devices, a 33 MHz Clock Generator (AT97SC3204 only), reset Switch, RC reset delay, decoupling capacitors, pull-up and pull-down resistors, and test points, among other things.

Benefits

- **Easy way to understand the operation of TPM devices**
- **Source code, hex images, TCG documentation and schematics all included**
- **Connects directly to PC via USB port**
- **Updatable code**



98. *Trusted Platform Module (TPM) Kits Selector Guide*

Name and Ordering Code	Function	Detailed Description	Kit Contents	
AT97SC3205T-SDK2	Development Kit for AT97SC3205T I2C TPM Device	Custom USB based development board based upon the Atmel SAM4S ARM microcontroller and **Atmel AT97SC3205T Trusted Platform Module (TPM) device with I^2C interface.** User-friendly demonstration and evaluation software is provided on a USB flash drive. The drive contains demonstration Source Code, hex images for reloading the demo code (if necessary), Trusted Computing Group (TCG) Documentation, and Kit Schematics. A USB extension cable is also included. The kit is updateable with the latest firmware, when available. (Please contact crypto@atmel.com for more details on demonstration software updates.)	• AT97SC3205T I2C TPM board (TCG v1.2) • USB extension cable • USB Flash drive	
AT97SC3205P-SDK2	Development Kit for AT97SC3205P SPI TPM Device	Custom USB based development board based upon the Atmel SAM4S ARM microcontroller and **Atmel AT97SC3205P Trusted Platform Module (TPM) device with SPI interface.** User-friendly demonstration and evaluation software is provided on a USB flash drive. The drive contains demonstration Source Code, hex images for reloading the demo code (if necessary), Trusted Computing Group (TCG) Documentation, and Kit Schematics. A USB extension cable is also included. The kit is updateable with the latest firmware, when available. (Please contact crypto@atmel.com for more details on demonstration software updates.)	• AT97SC3205P SPI TPM board (TCG v1.2) • USB extension cable • USB Flash drive	

\

99. *FAQs:* **Some important questions:**

Question 1:

I have AES cryptography already, so why do I need anything more?

If I am encrypting, isn't that all the security I need?

AES is an algorithm that scrambles (i.e. encrypts) information. The scrambled information can then be transferred somewhere else or stored in encrypted form and will remain unusable until it is unscrambled (decrypted) into its original form. AES requires a <u>key</u> to encrypt and the <u>same key</u> to decrypt. Both the transmitter of the encrypted information and the receiver of it must store that key and keep that key secret on both sides. If the key is not kept secret on either side then the information can be obtained by unintended outside parties, which defeats the whole purpose. So, securely storing the key is one of the most critical parts of security. The memory where the key is stored must be able to withstand attacks that try to read the key(s). Hardware security devices, like Atmel's CryptoAuthentication devices, offer a method of protecting secret keys that restrict access and provide key generation and management. Hardware security is much stronger then software based security because of the defense mechanisms that are on hardware security devices that repel attacks of various types.

Question 2:

I have heard that security has rankings, and that they rank something like this 1) SHA, 2) AES, 3) RSA, 4) ECC.

Does Atmel have the highest ranking?

In essence, security is a function of two main critical factors: 1) The length of the key used by the cryptographic algorithms, and 2) the mathematical operations that are employed by the cryptographic algorithms. The strength of security, therefore, depends on both the key size and the specific algorithms employed. So, any of these algorithms might be stronger or weaker than any other. Among most cryptographic experts, the following key sizes are considered to be approximately similar in strength: SHA-256, AES-128, RSA-3072 and ECC-256. The letters depict the algorithm (e.g. SHA) and the numbers after the dash represent the length of the key (e.g. -256). These important industry standard algorithms are all used in different ways depending on the application, and many cryptographic systems even use several types of algorithms together to

Question 3:

Symmetric algorithms like AES and SHA use a single key for all products (e.g. the host and the clients), so if the key is "broken" once, it would then be broken everywhere. Why would I want that?

The secret to understanding cryptography is that operations <u>always</u> depend on keeping secrets (literally), and that means <u>keeping the secret keys secret</u>. Secure storage of the secrets (i.e. keys) is what maintains a system's integrity <u>regardless of the algorithm.</u>

Fortunately, because of clever cryptographic engineering it is not necessary for all systems using symmetric algorithms to store the exact same key, even though this may sound counter-intuitive. Using a technique called "key diversification" each client in the system can store a unique key that is derived from its unique serial number and the key in the host (i.e. the root key). During the authentication process with diversified keys the client sends the host the client's diversified key and the its unique serial number. The host can then derive the root key from that serial number and that diversified key and compare that derived root key to the host's stored root key. If the stored and derived root keys match then the host knows that the client is real (i.e. authenticated).

Question 4:

How can I justify the cost of an additional device to put security in every system I produce?

Adding security into a system is similar to buying insurance. The cost of a security breach is analogous to the cost of an accident, fire, earthquake, or worse. The small cost of adding a crypto device ensures that future catastrophic costs are not encountered.

Some of the benefits of adding a crypto device are as follows:

1) **Increase revenue that can be lost due to the sale of unauthorized products (e.g. clones).**
2) **Improve revenue by preventing authorized subcontractors from using unauthorized sales channels (i.e. selling clones in the gray market).**
3) **Minimize the cost of warranty support for non-authentic (i.e. cloned) products.**
4) **Enforce product and manufacturing licensing agreements to gain royalty income as agreed to.**
5) **Increase product family revenue by allowing only authorized accessories to operate on an OEM's system.**
6) **Conform to regulatory requirements such as HIPAA for medical information privacy in the US, cross-border medical authentication recommendations in Europe, and new FDA recommendations in the US for wireless medical device authentication.**

Question 5:

How do I protect the bus between the crypto device and microprocessor?

The question often comes up about how to ensure that the communication between the crypto device and the host MCU is not attacked by a man-in-the-middle. In order to prevent a hacker from manipulating the bus between the two chips, the “Authentication OK” signal (i.e. the Boolean response) can be uniquely encrypted by the crypto IC.

One way to secure the bus between the crypto IC and MCU is to perform an authentication on the result of crypto calculation on crypto device (this is called the CheckMAC function). The idea is to make sure that when the authentication (i.e. crypto) device puts out the Boolean response (which means “True” or “False”, or one or zero) from the Check MAC operation that the response cannot be tampered with. The following paragraph explains in detail how that is done on a technical basis. However, the simple point here is that the Atmel’s CryptoAuthentication devices contain a built-in mechanism to prevent an attack on the bus between the device and the MCU by effectively encrypting that communication.

Technical Explanation: All the Atmel devices have a method of protecting the single Boolean bit that comes from the authentication chip to the microprocessor. It involves using the second key that is both stored in the CryptoAuthentication device and compiled into the code. After the successful completion of the “CheckMAC” operation, the second secret is copied into the TempKey register. Then the MCU sends over a unique number (for example, time of day), which is then combined with that second secret using SHA and returned to the MCU. The software on the MCU does the same combination using the compiled secret to see if it agrees with the result from the authentication device. This is beneficial, because it means that you cannot just put a simple switch in the wire between the two and always send a 1 (i.e. “True”).

The Atmel cryptographic ICs can also be set up to implement a secure boot with any MCU. In this mode the devices confirm the authenticity of downloaded code.

Question 6:

How can you encrypt with a one way algorithm like SHA, and isn't XOR weak?

Many encryption algorithms like DES and AES use the XOR function as a key part of their implementation. Those algorithms use a secret key to produce code that is XORed with the data in order to encrypt or decrypt that data. The secrecy of the encrypted data depends upon how well the crypto algorithm mathematically scrambles that code, and not upon the strength of the XOR function itself.

Hash algorithm-based devices (such as the ATSHA204) use the SHA256 algorithm in the same way to implement an encrypted transport of data between the crypto device and a microprocessor.

During an encrypted read operation the crypto device uses a random number that is hashed with an internal secret (i.e. the secret key) to create a result (i.e. a hash value or digest). That hash value (i.e. digest) is then XORed with the value to be sent out of the device. (In other words, to encrypt the plaintext, the plaintext is XORed with the hash of the secret key and a random number.) When that encrypted data is received outside of the crypto device, the receiver will need to know the secret (i.e. secret key) that was used in order for that receiver to reproduce the device's internal value that he or she will XOR with the received encrypted data to mathematically reconstruct the plaintext.

Similarly, on an encrypted write operation the chip inserts data based on the XOR of the internal calculated result. The encrypted data sent to the device should be calculated using the same internal secrets or the chip will not contain the expected value.

Since a completely new random number is created for each and every encrypted transport, and because the hash of that random number with the secret key is used as the XOR code, then breaking such an encrypted transport would require actually breaking the SHA-256 algorithm itself or obtaining the secret key.

Question 7:

I have heard that my system isn't secure unless I use a secure micro, is that true?

Secure Microprocessors are a great solution if your entire system can be implemented inside the provided resources of the secure micro and the cost point of the product is appropriate for the system.

For today's high performance systems, high-end microprocessors with external expandability or special purpose hardware are required for most solutions. So, secure microprocessors are typically used as a separate companion device to the system MCU. The security challenges then becomes two-fold: 1) Coupling the secure micro with the unsecure operations of the high performance MCU, and 2) Protecting the bus between the two.

The best choice today to address those two issues is using a specialized crypto device (like Atmel's CryptoAuthentication) or a secure micro.

A very important point is that a secure device is only as good as the protection of the stored secrets (i.e. keys) in a specialized crypto key storage device or in a secure MCU.

Special purpose Cryptographic IC devices (like CryptoAuthentication) focus solely on secure authentication and key storage, and as a result they can be the more cost effective choice, and often far more secure.

The security functionality of a dedicated hardware authentication device is carefully tested and verified by the manufacturer (e.g. Atmel), so the system designer does not need to worry about bugs that could be exploited by attackers. In addition, with a hardware approach, the system designer does not need to be concerned about future software updates of the software-based solution that can later expose a security weakness.

Question 8:

How do I protect my valuable software IP?

The only way to fully protect the system software is to use a secure (i.e. smart card based) microcontroller and store the entire system code within the device. In that way it cannot be read, copied, or modified by an attacker. However, due to cost and other concerns, this will not be an acceptable solution for most systems. On the other hand, a specialized cryptographic authentication IC can provide excellent protection at a low cost.

In order to stop others from copying the system firmware and producing duplicate products (i.e. clones), a crypto IC can be employed to match the firmware to legitimate products. Authorizing firmware can be done by sending challenges to the cryptographic IC regularly during normal operation and then checking for the correct response.   If a system is build doesn't include a correctly programmed crypto IC, the firmware will fail to operate, thus protecting against cloning.

If, in the above scheme, the firmware is updated on a regular basis and new challenge-responses are incorporated into the system, it becomes very difficult for a hacker to analyze the firmware and remove the authorization checks. If a remote connection is available, implementing some of the challenges randomly from a trusted backend server can make this nearly foolproof.

Processors like the Atmel ARM9 devices include an on-chip boot ROM. This ROM can be programmed to use a crypto IC to check a firmware signature stored in the external flash prior to executing that program. If unauthorized modifications have been made to the operating program, it will be immediately detected and system operations can be interrupted.

When system architects are looking to protect sensitive algorithm, data, or protocols, the same secure boot methodology can be used to decrypt the encrypted program module stored in the external flash. The decryption key can be stored in the crypto IC since there is often no secure storage in the system processor.

Question 9:

Is there licensing involved with using the Atmel devices?

The cryptographic algorithms used inside Atmel Crypto devices are industry-standard and approved algorithms that do not require paying a license fee to Atmel or any third party. In addition, Atmel offers communication layer source code free of charge for users to employ in any way without royalty.

Question 10:

Why would Atmel's chips be the preferred devices to use?

Atmel chips are designed from the ground up to provide an ultra-high level of hardware security. Atmel is now beyond the third generation of devices that contain years of high security device making experience and know-how. Some of the benefits that accrue to the Atmel devices are noted below:

- **Atmel uses proprietary test methods to eliminate test and probe access points**
- **Other chips use standard layout methods, but Atmel implements a serpentine active metal shield over the entire device preventing probing of internal secret nodes by providing a physical barrier and by detecting and acting upon attempted attacks.**
- **Atmel chips use the SHA-256 and ECC algorithms to ensure a long system life. CRC based systems cannot stand up to modern cryptanalysis and SHA-1 is no longer recommended for new system use by the US government.**
- **Few authentication chips include a random number generator, forcing the system to use weak methods to prevent replay attacks. Atmel devices include a multi-level FIPs level random number generator on-chip for the highest quality output.**
- **Atmel takes great care to analyze all possible input conditions to ensure that faulty or aggressive input sequences will never result in the loss of secret data.**
- **Many companies have the ability to design a straightforward digital device. But Atmel includes environmental tampers to prevent operation outside the specification range, encrypts internal busses to prevent emissions attacks, controls timing to prevent timing attacks, uses an internal clock generator to prevent "over-clocking", and provides many other security features.**

Question 11:

It seems hard to trust anything in the constantly changing digital world. How can we build IoT systems that can continue to be trusted over time?

Trust is probably the most important part of any IoT system, because unless the data is trusted it is of little real value. This should be obvious. In an IoT system the three pillars of security should be present: Confidentiality, data integrity, and authenticity. The key to all of these is ensuring that the secret keys and other important things stay secret. Storing digital IDs, passwords, secret keys, private keys, names, and other important private data in software alone is not a great idea because software has bugs, and can thus be hacked. Protected hardware-based key storage is the best alternative.

Question 12:

Constant change means constant software updates. How do you architect an IoT system to protect this update process?

1. **Protect the download image (don't let it be modified, don't let it be loaded into a clone system)**
2. **Repeatedly verify the code stored in the system (Flash), one way is secure boot**
3. **Use signatures to validate the code image, code fragments. Depends on an immutable public key, correct cryptographic implementations. Don't cheat on the algorithm or key size. Think about how to update or augment the keys in the future.**

Build a layered update process so that the unchangeable core can be as simple (and as well tested) as possible. Push as much as possible to as secondary boot layer that can be validated by the primary one.

Create a method which allows for flexibility in the download. Support broadcast images as well as (possibly partial) targeted images.

Questions designers should be asked:

- **Given the recently publicized attacks on unauthorized firmware modification, do you have a secure root of trust to validate your firmware image?**
- **How are the root private keys stored in your system?**
- **Can any software (and potentially malware) access, copy and misuse those keys?**
- **Do you have a validated implementation of your crypto firmware, or are you using open-source routines that may have limited validation?**
- **Are you using asymmetric algorithms (RSA, ECC) to create unique session keys?**
- **Are the memory and performance requirements a problem for your system?**
- **Do you have a cryptographic-quality random number generated, which is required for adequate security?**
- **Does it follow all the FIPS guidelines?**

Security
at our Core
Atmel Has You Covered

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370