



256 Bit Key – Is It Big Enough?

By Kerry Maletsky, Crypto Products Business Unit Director

Summary

Most cryptographic algorithms, devices or protocols are limited in security by the size of their key or other secret. This paper addresses the use of cryptography for the purpose of product authentication, whether it be a physical item or a logical block of firmware, and the key size necessary to be confident that it will not be guessed.

Authentication:

Development of products using the latest technology costs a lot of money. The greater the development cost, the greater the temptation to clone the product. Counterfeit goods comprise between 1% and 5% of worldwide trade, and it is growing at an alarming rate.

Since the cloner doesn't have a reputation to protect, quality and performance often suffer. The cloner can make a greater profit and offer the product at a lower cost by bypassing the development process and cutting corners on product safety and reliability. The result can be annoying if an ink cartridge fails. It can be expensive if a counterfeit battery damages the end-product. It can be life-threatening if a medical consumable is below standard.

Another issue is microprocessor firmware. Hackers are constantly figuring out ways to defeat product features intended to protect the end-user from unauthorized firmware downloads. While opening up a mobile phone to additional service providers may seem attractive, it puts the phone at risk of getting malware that could compromise the end-user's personal information or damage the phone itself.

The harm done to the end-user by counterfeit products or software can damage the reputation of the OEM, increasing product liability, maintenance and warranty costs, and decreased sales in the future..

How To Authenticate Something:

OEMs have always tried to protect their products and guarantee their authenticity in a variety of ways. There are many procedures that are in common use to authenticate items:

Source: If we trust the seller and we believe in the reputation of the shipper, then we might use this as a method of authentication. It is easy to see potential issues:

- Truly secure shipping, such as an armored truck, with 100% control over the item at every instant of the process, is quite expensive and rarely used. While uncommon, various kinds of substitutions can take place without even the shipper knowing.
- Even a reputable seller may not have complete control over their own supply chain – they may have been tricked in some way to offer counterfeit products under their own good name.
- We assume that some sellers, such as a sidewalk vendor selling \$100 baseball jersey for \$15, sell counterfeit goods. Yet the street vendor hasn't manufactured the items - what other sales channels is the counterfeit manufacturer using?

Physical Attributes: For items that have a unique physical shape, such as computer batteries, printer toner cartridges or even vacuum cleaner bags, we often just assume that if the label 'looks' authentic, carries the right logos and the product 'works', that it is authentic.

- It's hard to disbelieve our eyes – if the computer works, the printer prints or the vacuum cleaner cleans – that often serves to convince us, even

though we know that cloned versions are available and printing counterfeit labels is cheap.

- We may not ever find out the truth – how many people actually count the number of pages their replacement toner cartridge actually printed, or can tell what size particles are being passed right through that vacuum cleaner bag?

Holograms: This is common for clothing and many other retail items. The tags look good and appear to be hard to reproduce using equipment we are familiar with. But enter “hologram printing” on any search engine and you can find many companies willing to print these for you.

- A related confidence builder is a metalized label with a serial number embossed in it. But these are no more difficult to duplicate than a hologram.

Passwords: We are familiar with these to log into our computers or access accounts/information on a web site. But they are also often used internally in a controller-based system to validate a separate chip, board, consumable or network device. Or they may be used to control ‘special’ access to the system for configuration, maintenance, etc. Shared secret encryption keys are a close cousin of passwords, with the same advantages and disadvantages:

- There is usually no place to safely store the expected password on the receiving system, so if an attacker has access to that system he/she can extract that value from the EEPROM, FLASH or other nonvolatile chip.
- Usually, it’s easy for an attacker to find a way to ‘watch’ the system when it’s in use and discover the password as it is being passed from one system element or another. Or the attacker might record the entire session and just replay it at a later date to repeat the benefit the first user obtained.
- A somewhat more complex version of this kind of security is to include a public key signature of the serial number in a consumable. The host device can validate the signature without having to store any secrets. But since both the serial number and signature are typically stored in an unsecure device, they can easily be read and copied into the clone device.

Smart Card: Some satellite, cable or other media providers use a smart card to authenticate a user to the network. The user plugs the card into the set top box in order to make the media available. These are often very secure but they are not perfect for all applications:

- They may be physically impractical if the device is too small, is used in a wet or dirty environment or if the end use (many consumer items) mandates no parts to lose.
- Smart cards can cost \$5 per unit, way too much for an end-product priced under \$100.
- The total system cost is even higher since a physical reader slot and connector has to be supplied, in addition to the cost of the electronics to support the ISO 7816 interface and the cost of the card itself.

Low-cost, Cryptographic Authentication Technology to the Rescue

Hardware authentication devices have been available for some time, but only recently have they been able to combine proven & robust cryptography with low cost and ease of use in the typical microprocessor-based application. Virtually all security devices contain some sort of secret along with a cryptographic processing element. Generally, the secret can never be read from the chip, rather the secret is combined with input data using a protocol that proves the knowledge of the secret without revealing it.

Cryptographic chips with wired or wireless interfaces are available with increasingly impressive capabilities that can make the counterfeiter's task very difficult indeed. Wired devices may be soldered down on a board with other system components, or may be attached to a consumable and connected to the system via contacts. Wireless RFID chips don't require contacts and are optimal when the environment is challenging.

Usually, these authentication chips incorporate a serial number and offer several advantages over any other kind of serial number storage:

- The number can be changed, since it is programmed into the silicon by the chip manufacturer.
- The serial number can be cryptographically connected to secret keys on the chip which cannot be read or copied. The attacker needs both the serial number and the secret key to build a counterfeit device.
- The chip can provide a way to combine a dynamic, random challenge from the host with the serial number – a much better solution than the static signature mechanism discussed above which is susceptible to copying.

The CryptoAuthentication™ family of chips from Atmel is the newest of this breed of devices. Incorporating security features developed from a long history of security chips it provides an unprecedented combination of security and ease of use – at a cost that is lower than that of existing authentication chips.

These chips use the SHA-256 hash algorithm to avoid any known algorithm weaknesses. In addition, they incorporate a full active metal shield over the entire internal circuitry – if an attacker cuts or shorts any wire in the shield, the chip stops functioning. Added to this are internal clock and voltage generation, fully encrypted memories, tamper detection and fully secure production test methods.

Modern processing technology allows the chips in this family to be incorporated in a SOT23 package which is less than 2mm x 3mm – tiny enough to be incorporated in the most space constrained portable systems, incorporated inside battery packs or fit on existing PC boards without even increasing their size.

All members of the CryptoAuthentication include a 48 bit serial number which is guaranteed to be unique, along with the appropriate cryptographic protocol to validate that the number is not a simple copy on a counterfeit product.

A single wire interface simplifies the mechanical connection to the device while reducing the number of GPIO or UART resources required on the host microcontroller. An automatic sleep mode reduces the standby current to less than 100nA when the crypto operations are completed. And the straightforward challenge-response mechanism of the CryptoAuthentication devices, along with the use of an algorithm that is widely supported by commercial and open source software libraries simplifies the programming requirements.

Two important characteristics of every cryptographic chip are the size of the key and the strength of the algorithm. It's pretty easy to imagine that bigger keys are better – but just how big is big? And while it's tempting to think that the newest secret algorithm is the best, “security through obscurity” is generally considered to be very risky. Crypto experts much prefer algorithms that are well publicized and have been analyzed by lots of smart people over years. The following sections discuss these concepts in more detail.

Is a 256 bit key big enough?

As computational ability rapidly increases, more concern is being placed on the key size in cryptographic devices. Individuals commonly have a quad processor 4GHz computer on their desks, so trying billions of possibilities to crack a secret key is pretty easy. These attacks are usually called “offline” attacks since the attacker doesn't use the host or client system to try each possibility. Instead he uses external computers to mimic the computation of the authentication chip to guess the stored secret, trying to generate a sequence of bits which matches that which was recorded once on the authentic system.

In the simplest example, a “brute force” attack, the attacker gets a complete or partial clear text message and the corresponding version of the message encrypted with the key he wants to crack. He then successively tries each possible key until he finds the one that creates the correct encrypted message. If there are n bits in the key, then after 2^{n-1} attempts the attacker has a 50% chance of finding the right key and after 2^n attempts he has tried all possible keys and is guaranteed to have found the key.

The only protection against such a brute force attack is to choose an algorithm that uses a key so big that it will simply take too long to try a very large percentage of the possibilities. Keys that were big enough 10 years ago are not big enough any more because of the exponential growth in computing power. Here are some well publicized successful brute force exploits:

An array of 64 Virtex-5 FPGAs from Xilinx can successfully find a 48-bit key in less than an hour. (See http://www.usenix.org/events/sec08/tech/full_papers/nohl/nohl.pdf) The Mifare® cryptographic memory chip used widely around to protect electronic purse contents uses 48-bit keys.

The official encryption standard adopted by the United States in 1976, Data Encryption Standard (DES) uses a larger 56-bit key. Several machines have been built that can find a key through brute force in less than a week. (See http://en.wikipedia.org/wiki/EFF_DES_cracker)

Although no successful brute force attacks have been reported for commercial chips using algorithms with key sizes greater than 56 bits, it is expected that algorithms with larger key sizes will eventually become vulnerable with increasing computational ability. As of the writing of this paper, the US Government is recommending Advanced Encryption Standard (AES) with a 128 bit key for government encryption purposes. Setting aside any mathematical weaknesses in AES (if they exist), this means that the government believes that attacks against a key space this large will be impractical for some years to come. However, with computing power doubling every 18 months or two years (See http://en.wikipedia.org/wiki/Moore%27s_law), 128-bit keys will eventually become “crackable” using brute force attacks.

As a result some system designers look for even larger keys to ensure that a system they design today will still be secure during its entire life – even after much larger and faster computers are available to hackers. A key size of 256 bits is so big that all cryptographers agree it is immune from exhaustive attacks. Just how big is 2^{256} ?

Here are some estimates of big numbers:

2^{66}	Number of grains of sand on the earth
2^{76}	Number of stars in the universe
2^{79}	Avogadro's number. The number of carbon atoms in 12 grams of coal.
2^{96}	Number of atoms in a cubic meter of water
2^{190}	Number of atoms in the sun
2^{255}	Number of attempts to find the key in this chip

But what about very well funded entities such as the US National Security Agency (NSA)? Could they build a machine to crack a 256 bit key? Assume they could build a theoretical nanocomputer that executes 10^{13} instructions per second (approximate rate of atomic vibrations) in a space of a cube with a side that is 5.43nm across (This is the approximate size of a silicon lattice 10 atoms wide, or a crystal containing 1000 silicon atoms). Assume that it could calculate an attempt in 10 cycles. Such a computer the size of the earth would take more than 10^{13} years (roughly 58 times the estimated age of the earth) to attack a 256 bit algorithm via brute force.

Is a 256 Bit Key TOO Big?

There are a few downsides to larger keys – they increase the complexity of low cost authentication chips in a number of ways:

More internal memory storage to retain keys and temporary values. Usually the largest blocks on most such chips are the memory arrays. Doubling the size of the keys typically doubles the total amount of nonvolatile and volatile data memory which could therefore increase the chip cost. However as the line widths in chips

shrink, the core size of the memory cells becomes a smaller and smaller percentage of the total chip area reducing this cost penalty proportionally.

More logic gates and hence larger, more costly chips. It's generally reasonable to assume that doubling the size of the key will double the size of the logic to implement the block. Alternatively the same logic size could be used at a penalty of perhaps 2-4x the computation time, depending on the algorithms in question. Implementing the chip in a newer technology with smaller transistors can offset this disadvantage.

More transmission time. Typically both the challenge and the response are the same size as the key (if not, then the shortest of the three can be attacked more easily than the other). So doubling the key size will double the transmission time for the transaction. But since authentication is done infrequently (perhaps on power up only), this penalty matters less in the overall scheme.

Cryptographic professionals (and hackers) are a creative bunch. Even though the time scales in the previous section seem daunting, new attack procedures could be found that might simplify the task by a factor of 2, or 2,000 or 2,000,000. Increasing the search space with a larger key helps to ensure that even with these advances, it will remain extraordinarily difficult to guess a 256 bit key anytime soon.

Why not just keep the hash algorithm secret?

If the attacker doesn't know the algorithm, then implementing a brute force attack is impossible since the attacker can't compute the output even if he knows the key. Systems like this were the historical norm until very recently.

This is still a reasonable strategy in some situations, especially where there is a limit on the complexity of the encryption hardware (perhaps for cost or power consumption reasons) and/or insufficient key storage mechanism. Good examples of this situation would be RFID tags which cannot consume very much current nor cost more than the value they protect, perhaps a single trip on a subway.

Nonetheless, such systems are being used less and less in favor of systems constructed from widely studied open algorithms. This has been made possible by advances in semiconductor technology that permit logic gates to cost less and consume less power at the same time.

It's very hard to maintain the secrecy around algorithms:

- The German WW2 Enigma machine was secret only until one was captured by the Allies and its weaknesses were uncovered by clever mathematicians
- The encryption algorithm originally encrypting European GSM cell phone conversations was protected by a non-disclosure agreement (NDA) until a

university accidentally disclosed it without getting a signature on an NDA. It was promptly broken and the attack published.

- The encryption algorithm in the Mifare chips (see above) was teased out of the logic on the chip by another university team that legitimately purchased devices that implemented the algorithm. They studied the logic under a microscope to find out how it worked.

Better hardware design strategies that include countermeasures for historical and anticipated security attack methodologies can increase the useful life of systems with secret algorithms further into the future.

On Hash Algorithms

Cryptographic hash algorithms are designed to convert or compress a variable length **message** into a fixed-length string called a **digest**. If the digest uniquely identifies the message then the digest can be used as a stand-in for the message, shortening the computation time for various operations. While many algorithms can be used for simple hashing functions, a cryptographic hash algorithm has a few important properties:

1. It should be very difficult to find two messages for which the digest is the same. If two such messages do exist, this is said to be a collision.
2. Given a particular digest value, it should be very difficult to create a message that would produce that digest.
3. It should be relatively easy to create the digest from the message.

Digests can be used to verify the integrity of the message by performing some cryptographic operation using both a secret key and the message digest, the output of which can be used for message validation. If the recipient of this validity code has the knowledge of the secret key, he or she can be confident that the message sent along with the code was not modified while in transit.

When used in this way, such a validity code is usually called a message authentication code (MAC). Usually we say that attached to a message is a MAC, which was generated using both the message and a secret key. The MAC algorithm is considered to be strong if it is very difficult for the attacker to create message/MAC pairs without knowledge of the secret. As well, it should be impossible for the attacker to change the message in a way that the same MAC would match it. Hash algorithms are often used to implement MAC algorithms.

The SHA-1 and MD5 hash algorithms have been widely used for cryptographic purposes. However, recent mathematical analysis has shown that there may be weaknesses in these algorithms. As a result, they have been replaced in the suite of algorithms recommended for use by the US Government with the SHA-2 family. The best known of the algorithms in the SHA-2 family is SHA-256.

The typical attack on hash algorithms is to find a collision – two messages that hash to the same digest. The reason for this is twofold:

1. If the hash algorithm is being used as part of a message authentication or signature scheme, then the attacker can create one message that the sender authenticates but substitute the other message which the recipient will believe to be authentic. This would have significant benefits for the attacker – if the attacker could change the shipping address on an order, for instance, he could receive the goods without paying for them.
2. Due to the Birthday Paradox (http://en.wikipedia.org/wiki/Birthday_paradox) this kind of attack takes far less attempts than is obvious. If the digest has n bits, then only $2^{n/2}$ random messages need to be hashed in order to find a collision. This is the same as cutting the number of bits in half!

As a result, cryptographers have put a great deal of effort into finding ways to create two messages that collide. For SHA-1, while the expected strength against attacks would be to require 2^{80} attempts, the current state of the art attacks require only 2^{63} attempts, potentially within range of a brute force attack.

The Birthday Paradox requires that the attacker randomly select pairs of messages. This seems to limit its usefulness, but an example with an email message shows why it's very powerful. We can't see space characters at the end of a line in a simple text email, but there may be a variable number at the end of each line. If the message is relatively long it's easy to see how a huge number of messages can be generated each of which appears identical but all of which are actually unique. A similar concept can be used with an image attachment – two images may appear to our eyes to be identical but may in fact be very different at the bit level.

The brute force birthday attack does not work on most authentication devices, however, for a few specific reasons:

1. The usual implementation of the birthday attack is to compute digests of $2^{n/2}$ versions of the original message all of which appear to be the same and $2^{n/2}$ versions of a beneficially fraudulent message and compare all of the digests in the first set with all the digests in the second set. Since in authentication chips all the bits of the original message are known to the verifier (the message is short and of a fixed format), the first set can't be created which forces the second set to be 2^n in length.
2. Incorporating a unique nonce into the message prevents pre-computing digests of a large array of messages that might be compared to those recorded during each of many authentication operations. This is because each 'correct' message contains a shared element (the nonce) that is different from all previous 'correct' messages. Care must be taken to ensure that nonces are not reused, however.

Some attacks to find collisions are made easier by being able to vary the length of the message - the fixed length message property of authentication chips can inhibit these. This property also inhibits "length-extension" attacks, in which an attacker can 'extend' an unknown message with a known value and create the proper digest for the new extended message. Combining a hash algorithm with the HMAC construction also prevents length extension attacks.

Is The Latest and Greatest Algorithm Sufficient?

Attacks on the algorithm itself can be prevented by using the latest and greatest algorithm. But this is not enough. While authorized systems interact with these security chips according to the datasheets, the attacker has access to a whole range of options well outside the normal operating range, up to and including removing the package from around the chip and analyzing the very components within the chip.

Since the purpose of these algorithms is to prevent the security chip from having to reveal its stored secret key in the clear the algorithm must be combined with a whole range of additional protections to ensure that the secret cannot be obtained by means other than a cryptographic attack.

- Physical protection against attacks. Equipment to probe internal nodes of operating chips is widely available, so authentication chips should include active shields to cover internal nodes, use the latest processing technology to reduce the size of the internal nodes and incorporate multiple layers of internal interconnects, preferably more than 3 layers. All these make microprobing more difficult
- Secure cryptographic protocols. Most algorithms have known weaknesses if used in an improper way so the way in which the chip uses the algorithm must facilitate its secure use. Since an attacker can usually record every bit going back and forth between the chip and an authentic system the protocol must provide anti-replay protection.
- Environmental extremes like fast clock rates or supply voltages that violate the datasheet can often cause a chip to malfunction. In some cases this malfunction can permit the secrets to be read from the chip. State of the art secure designs prevent this from happening by controlling the environment or detecting extremes and shutting the chip down.
- Improper command or IO usage. Many programmers are familiar with stack or memory overflow attacks against some systems which occur when some function is presented with extraordinarily large inputs or passed an illegal value. Well designed security chips are specially constructed to carefully analyze every input and reject all but those which are acceptable.
- Information leakage. Beyond the expected IO channel there are other ways in which information may pass from the chip to the attacker. Perhaps the timing of an operation indicates something about the internal secret. An attacker can measure the current flow into the chip over time to see if there is an unusually large or small current flow for one condition or the other. Sometimes there may be some sort of electromagnetic emission that can be measured. While no chip can provide perfect protection against every kind of known or unknown leakage, security chip designers are familiar with these attacks and can provide a significant level of protection.

Summary

There are broad reputation, safety, liability and profit reasons to incorporate hardware authentication in all new designs. High quality authentication solutions are available to protect a wide range of things from cloning, fraudulent modification, secret disclosure or other types of misuse. The elements being protected can include software/firmware modules, media files, medical consumables and records, electronic consumables such as batteries and printer toner cartridges, other retail consumables such as filters, and wireless or network transmissions, just to name a few.

So long as the device or host device contains some sort of microprocessor or host computer, modern authentication chips can be used to bring a level of security to the design that was never before achievable. The availability of proven cryptographic algorithms simplifies the implementation so the designer doesn't have to be a crypto expert.

When selecting an authentication solution for products the designer needs to find the right balance between cost, security, and speed for his/her application. In addition, the designer should consider the lifetime of the product in the market to ensure that the authentication mechanism will still be secure at the end of the useful product life.

For applications that don't require lots of memory storage or different cryptographic protection for different sections of the chip memory and don't require multiple algorithms on the same chip, cryptographic authentication ICs can offer the highest level of security at prices that make them appropriate in most mass markets.

While Moore's law dictates that the counterfeiter will have access to progressively cheaper and faster computation horse power with which to build the clone device or crack the keys, it also means that high security chips with progressively greater security and lower cost are available to the legitimate OEM. In the case of key size, bigger is always better.

Editor's Notes About Atmel Corporation

Atmel Corporation is a global leader in the design and manufacture of innovative integrated circuits, focusing on microcontrollers, ASICs, nonvolatile memory, secure products, radio frequency components, and touch sensing technologies. Leveraging one of the industry's broadest intellectual property (IP) portfolios, Atmel provides electronics systems manufacturers with complete system solutions. This enables its customers to lead the markets they serve with electronic products that are smaller, smarter, more cost-effective and versatile than ever before. Target markets include automotive, industrial and medical electronics and secured systems such as smart cards, payment terminals, electronic ID, secure data storage/transfer and RF identification, as well as a wide range of latest-generation consumer products.

Further information can be obtained from Atmel's Web site at www.atmel.com.

© 2009 Atmel Corporation. All rights reserved. Atmel®, Atmel logo and combinations thereof, Everywhere You Are®, CryptoMemory®, and CryptoRF® are registered trademarks, and CryptoAuthentication™ are the trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.