Keywords: cryptography, algorithm, secure micro, cia, financial terminal, HSM, hardware security module, secret key, ibutton, smart card

TUTORIAL 5421

Cryptography in Software or Hardware - It Depends on the Need

May 25, 2012

Abstract: As the length of software keys increases to accommodate evolving needs for greater security, so the marketplace demands a wider variety of cryptographic implementations. With recent improvements in core design and frequency performance, designers are now asking whether customized IP blocks are still needed for these secure algorithms. In short, can a designer use a generic core in the hardware to save space and cost, and embed the cryptographic algorithms in software? The answer is simple—well, not so simple—it depends on the need.

A similar version of this article appeared on Embedded.com, August 2011.

Introduction

Cryptographic algorithms are high-performance, secure engines that require considerable space in a design. When countermeasures are added to thwart security attacks, the space and memory requirements grow even more demanding. For these reasons, cryptographic algorithms have traditionally been embedded as proprietary designs (i.e., intellectual property, IP) in hardware on smart cards or 8-bit chips. With recent improvements in core design and frequency performance, designers are now asking whether the customized IP blocks are still needed for these secure algorithms. In short, can a designer use a generic core in the hardware to save space and cost, and embed the cryptographic algorithms in software? The answer is simple—well, not so simple—it depends on the need.

The Science of Secrecy

Cryptography was originally designed and known as the science of the secrecy. It was the weapon of kings, generals, spies, and ambassadors. In the last century, cryptography has grown up to become a more sophisticated toolbox which provides information trust to its users.

Cryptographic devices have a long history, longer than the history of microprocessors. (As a reminder, cryptography and its dark side, cryptanalysis, is the origin of computer technology, thanks to the Colossus project.[1]) Military forces and diplomats developed cryptographic machines to protect their communications. Depending on the required strength (i.e., a tactical communication on the battlefield has a shorter lifetime than a diplomatic message), the machines met several constraints: speed, reliability,

integrity, protection of secrets, ease of use, and acceptable costs.

When security needs spilled over into the civilian world, new technologies and constraints evolved. Asymmetric cryptography[2] particularly soared for these markets while cost became a bigger concern. Meanwhile, trust, authenticity, and credibility were always paramount considerations for the financial and banking markets. Remote financial transactions would only be possible if cryptographic mechanisms could replace the traditional face-to-face agreement and handshake.

So the earliest cryptographic devices were strictly dedicated to security. With a form factor similar to a floor safe, evolving hardware security modules (HSMs) securely hosted and handled most secret keys. Made of steel and heavy, HSMs were, and still are, quite similar to military ruggedized boxes. These security modules consisted of a secure box and processed secure keys, mainly for authentication but also for keys generation, PIN codes generation, and key protection (storage). Security standards, such the NIST FIPS 140-2 certification program, the EMV® standard for smart-card use, and Common Criteria (ISO 15408) were developed and then accelerated the deployment of these devices.

Confidentiality, Integrity, and Authentication

The acronym CIA (which stands for confidentiality, integrity, and authentication) symbolizes the main services that cryptography can offer today. In the simplest terms, it gives an accurate answer to well-identified threats.

Confidentiality is required to avoid eavesdropping and unauthorized access to sensitive data that owners only want to share with authorized people. Thus, the embassy sends notes to its foreign ministry; the spy transmits results of actions and nobody else shall read what is written down ("for your eyes only"). In a world of consumers, confidentiality is required for transmission of pay TV programs to be accessed only by authorized subscribers. It is needed to protect personal data transmitted over public networks that are otherwise easily accessible. Mobile phones and Wi-Fi® connections are typical examples.

Integrity protects unauthorized and or uncontrolled modifications. Modifying devices means changing behavior (the software has been modified) and the resulting loss of confidence (the data are no longer trustworthy). This type of malicious modification is typically done with malware run on an otherwise trustworthy device while the device owner is misled. Banking data is particularly sensitive to this threat, as data integrity is lost if a financial transaction is maliciously modified.

Authentication guarantees the origin of information. How valuable is information if the source is not validated? Clearly the answer depends on the source and sensitivity of that information. A signed contract commits the signer, so verification of identity is as important as the contract content itself. Modern mobile devices and PCs accept only authenticated, authorized application updates and/or modifications. This ensures that no one takes control of the device and runs unauthorized software. Authentication protects a device or a component against counterfeiting. A trusted element can not only prove its origin, but it can also authenticate the origin of the device to which it is attached. Thus a battery, a printer cartridge, or consumable shows their "credentials" to the hosting device which, in turn, authenticates, "trusts," that consumable.

Cryptographic Algorithms Come with Costs

The crypto community has developed many algorithms to ensure these secure services. Asymmetric

cryptography is based on number theory and symmetric cryptography is based on more pragmatic "recipes." Because of their wide adoption and publication, these algorithms are available today as standards that are commonly and easily measured and scaled to the application.

The algorithms have a distinctive drawback, the consequence of providing an accurate answer to identified threats: cost, or more exactly, costs. Cost is measured in terms of execution time, memory use, power consumption, and die surface. These are not minor, insignificant issues; they are known to have forestalled or even prevented implementation of efficient security measures in devices in the past. Proprietary algorithms and scrambling algorithms are design shortcuts, used routinely in the industry and considered sufficient. These shortcuts admittedly consume fewer resources or simplify security management. Considered better than nothing, these solutions do not really deter attackers. Instead a professional approach to security promotes a reliable, standardized, and appropriate defense against threats that have been objectively identified.

The security industry has long understood these principles. All its history demonstrates the constant struggle between the trade-offs of lowest implementation cost versus the required level of security. An obvious example is with asymmetric cryptography (e.g., RSA, elliptic curves) where keys lengths correspond to the level of security; the longer the key is, the stronger the protection. But yet another rule applies in this context: the longer the key is, the higher the complexity. Restated, the computation is longer and the required resources are larger. A prudent risk assessment estimates which key length is sufficient for the task and, therefore, what is the minimum level of complexity required for the implementation. In fact, today the RSA keys used on simple devices are lengthening very slowly, while the devices' computing power increases. In the end, using too short a key weakens the implementation; using too long a key is needlessly expensive.

Evolution of Embedded Cryptography

Cryptographic algorithms are most commonly used in smart cards. The chip ensures security by checking cardholder credentials (typically a PIN code) and performing cryptographic operations.

Very simple in term of architecture, these cards were initially embedded with the algorithm and an 8-bit core, running at few tens of megahertz. The embedded algorithm provided a few services, but absolutely could not perform any cryptographic computation. For that reason, cryptographic hardware blocks had to be added.

Initially, the cryptographic hardware blocks only performed symmetrical cryptographic operations, because the algorithms were less complex than the asymmetrical counterparts. These hardware blocks, therefore, presented a strong security limitation as they required careful management of card personalization, the keys distribution, and the keys diversification. Moreover, the inability to perform on-board RSA computations led to security weaknesses. Consequently, the EMV static authentication scheme was the only method used for card authentication. Because those cards were not really so "smart," that scheme was quite prone to attack from fraudulent cards.

Hackers continued to exploit the weaknesses of the RSA hardware blocks while the costs of that hardware also dropped. More secure authentication schemes based on on-board RSA computation emerged. Nevertheless, die costs still limited the length of the used keys and the core still could not perform complex computations.

Other form factors appeared in the 1990s. Maxim's iButton® devices enhanced security by proposing an

innovative approach to keys protection and by making the platform more flexible. PCMCIA cards were also very popular for their easy acceptance in PCs, their fast data exchange rates, and larger casing.

Despite all these advances, as late as 2003 the architectures for cryptography remained the same: a slow 8-bit core, typically an 8051, backed by cryptographic hardware blocks. Other limitations remained: small RAM sizes (usually hundreds of bytes), and applications limited in scope (mostly in ROM code, especially for smart cards). The later emergence of nonvolatile memories such as EEPROMs and flash added flexibility for some applications, but did not significantly change the cryptographic conditions or methods of application.

Redesigning for Evolving Security Needs

For applications where security is a lower priority and where devices are less focused on cryptographic/security needs, a software implementation is usually the panacea. A bootloader that checks integrity and verifies the authenticity of the embedded firmware is commonly used today to prevent most threats to consumer and industrial devices. The fundamental cryptographic needs for these less-complex secure applications are hash functions and digital signature verification, and all are solved with software implementation. The software implementation is clearly simpler than embedded protection and does not impact the die size. It does not require a specialized hardware block that may not be used by every customer, and, finally, does not raise system costs.

Of course, applications evolve. If security was not needed yesterday, it might become crucial tomorrow. With no cryptographic blocks embedded in the existing solution, only cryptographic software and subsequent updates will provide the cryptographic upgrade. If the security need is high and must be embedded in hardware, then device redesign is required and that introduces potential solution breakdown.

A typical example of this process is energy metering, an industry where standardization is not complete and there is still considerable latitude for proprietary solutions and initiatives. Adding a security chip to an existing meter may be too complicated (i.e., an additional chip and undetermined PCB modification), too expensive, and thus too premature, compared to adding simple cryptographic services with software. All of this might not even be needed if only used sporadically at certain times in the meter's lifecycle. A cost-versus-risk analysis would quantify the value and payback to the meter's manufacturer and user. Historically, if justification for added security in mass markets is not clearly apparent, security has always been rejected for its extra cost.

Cryptographic Software Introduces New Risks

It is time to admit that adding security through cryptographic software does come with its own risks. When an algorithm execution is performed in an environment with other applications running concurrently, then information leakage from timing measures or timing variations or from cached data can allow those other applications to retrieve information on handled secret keys.[3] This threat is serious and any "open" platform is subject to it. Only authenticated platforms, specifically only platforms running authenticated applications, should be considered acceptable for a design.

Understanding the Trade-Offs: Cryptographic Hardware or Software Implementation?

You might be asking yourself whether to use a software or hardware implementation to get the security that you need. The answer depends on the need: how much security does an application really require?

There are still plenty of situations where implementing security with cryptographic hardware makes the most sense. When large volumes of secure data need to be processed as with pay TV or SSL accelerators, high-performance security can best be achieved in hardware.

If the security is implemented with a fast microcontroller, then the data will process faster than with an external software implementation. Hardware blocks can also simplify the design and avoid the problem of data processed by the core. Processing data simultaneously while the application is executing provides better real-time system performance. Recall, finally, that a hardware implementation costs more than a software implementation.

To complicate the subject further, note that, the trade-offs between a hardware or software implementation can exist in the same application. In this context the ultimate design decision should be based on the different intended uses of cryptography. A symmetrical, bulk encryption need might be addressed in hardware while the authentication, keys exchanges, and protocols setups are best addressed by software. Ultimately, the decision on how to setup the cryptography will be based on time, space, cost, or a balance of all.

There is another very important situation to consider. If an application will not be aware that a cryptographic mechanism is in use, then the cryptography must be in hardware. The most typical example of this situation is automatic encryption/decryption on the fly for program security. Only cryptographic hardware can transparently address such service, as it allows easy software porting and execution from an unsecure context to a very secure one without extra cost. Assuming that the encryption engine configuration is done, everything is performed transparently. Such embedded cryptographic solutions are present on several chips, ranging from the MAXQ® family of RISC microcontrollers to the DeepCover® Secure Microcontroller (USIP) platform. There are major benefits of an embedded solution like these devices: software development is not burdened with the need to develop cryptography, and even complex software like a full Linux® OS can be easily ported to a secure platform without code modification.

Some traditional situations favor hardware cryptography. Chips with very small cores (as in the "ancient" age of ICs) do not have the resources or timing to perform cryptographic computations in a reasonable manner. This is true even considering very optimized implementations. An RSA encryption can be considered feasible on almost every platform, while RSA decryption is totally unrealistic for keys lengths greater than 1024 bits.

When there is no room for software execution because the chip does not embed any core, cryptography in hardware is the only solution. This situation is typical with consumer products where costs are under pressure and security is not a selling advantage. In fact, this is where and how manufacturers want to protect their IPs against counterfeiters or competitors. While this defensive feature is not claimed or promoted to the customer, it is nevertheless present at lowest possible cost. Keeping those costs low means the smallest die size, which implies that the chip functions are minimized. We usually only see

authentication chips in these situations. The chips perform symmetrical cryptographic computations which involve a challenge and response: sending a secret and receiving verification of identity back. Symmetrical challenge-and-response cryptography is now routinely used with batteries for laptops and mobile phones because manufacturers are greatly concerned about counterfeiting.

Random number generation, if not pseudorandom, needs a cryptographic implementation in hardware too. The quality (i.e., genuine randomness) and ease of use of random numbers, (i.e., no need to store a state number) mandate hardware resources. Ideally software postprocessing can help produce more numbers by applying one-way functions.

Financial Terminals Require Higher Security

Recent standards like EMV, MasterCard® PayPass™, or VISA® PED have led to the development and improvement of new categories of secure devices. Today whenever one speaks of a smart card, you really mean smart-card reader. Authentication by PIN code means protection for that PIN code, and finally, PIN code encryption means cryptographic services are available.

One of the largest markets for such heightened security is financial terminals, the point-of-sale (POS) and automatic teller machines (ATMs). Although POS machines and ATMs were already using magnetic stripe technology, implementation of the EMV standard surged recently because of strong support from banking and credit card schemes. From several domestic U.S. markets, POS machines and ATMs have spread into a global, standardized market. Services related to payment transaction are growing because the POS and ATM devices now host rich and multiple applications. In fact, these business devices now function more like consumer devices, because they are like older PCs in term of computing power and they mimic smartphones by recycling their technologies (e.g., touch screen, color, portability, design).

Nevertheless, unlike the vast array of consumer devices, financial terminals still require a high level of security. These terminals are critical targets for tampering, so security for the standard PCI PTS addresses must include physical protection and broad use of cryptography.

A smart-card architecture is no longer the only choice for protecting these financial terminals. Now two major approaches coexist, the bichip architecture and the monochip architecture.

The bichip architecture combines a multipurpose, generic, 32-bit powerful microcontroller and a smaller companion security chip. This architecture enables the manufacturer to separate security services and other services; the security chip processes the security services while the other services are addressed by the generic microcontroller. The security chip may have sufficient computing power, but it usually contains cryptographic hardware blocks. This configuration is used mainly for historical reasons because quite often these chips inherit technologies based on forerunner designs like smart cards or iButton devices. This approach lets you add cryptographic services with an off-the-shelf chip to the inherited certifications of the older technologies already designed in. When this "smart" software operates and exploits the hardware blocks, users can implement almost all usual algorithms. As a result, the flexibility of the software and the performance of the hardware are leveraged together for maximum results.

The monochip architecture uses a single chip with a powerful 32-bit core, clocked at hundreds of megahertz, capable of running a rich OS with graphical effects and of handling various communication links (e.g., Ethernet, Wi-Fi, mobile). This chip addresses all the cryptographic tasks and makes sense because it can leverage the vast resources of the software for several obvious benefits.

- Flexibility in choice of algorithm; adding a new algorithm to the platform is as simple as a software update.
- Satisfying performance corresponds to the application needs.
- Large key lengths are supported without limitation, thus lengthening the secure device's lifetime.
- Addition of simple countermeasures.
- No extra cost on the chip; there is no die increase due to cryptographic use.

An example of such a secure single chip is the Maxim USIP PRO, which runs software implementations of the most common cryptographic algorithms (**Table 1**) very efficiently.

Table 1. USIP PRO Cryptographic Performance

Algorithms	Speed
SHA-1	2083kBps
RSA 2048 CRT decrypt	400ms
RSA 2048 encrypt	18ms
ECDSA P-192 sign	23ms
ECDSA B-163 sign	16ms

This performance demonstrates that the usual cryptographic needs, like those expressed in PCI PTS requirements, are addressed in an acceptable time.

Conclusion

As the length of software keys increases to accommodate evolving needs for greater security, so the marketplace demands a wider variety of cryptographic implementations. New algorithms (elliptic curves) may reduce application complexity, but still designers must wrestle with the trade-offs between implementing security in software or hardware. Although there will always be exceptions for specific services, niche markets, or very small chips, embedded cryptography is replacing software cryptography for a wider range of services. The very high level of security for financial transactions is a timely example of why and how only embedded cryptography can protect both those systems and their users.

References

1. For more information on the Colossus project and code breaking in World War II, go to http://www.tnmoc.org/explore/colossus-gallery.
2. Asymmetric cryptography is a system in which encryption and decryption keys are distinct and moreover, the encryption key can be public. Asymmetric cryptography is the opposite of symmetric cryptography, in which encryption and decryption are performed with the same key that remains secret.
3. Lawson, Nate, "Side Channel Attacks on Cryptographic Software", **IEEE® Computer Society**, Nov/Dec-2009). (www.computer.org/portal/web/csdl/doi/10.1109/MSP.2009.165).

Related Parts	
USIP	DeepCover Secure Microcontroller with MIPS 4KSd Processor Core

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5421: http://www.maximintegrated.com/an5421
TUTORIAL 5421, AN5421, AN 5421, APP5421, Appnote5421, Appnote 5421