Performance Comparison of Elliptical Curve and RSA Digital Signature on ARM7

Nikunj R. Nomulwar [1],
Dept of computer technology
Veermata Jijabai Institute of Engg. & Technology
Matunga, Mumbai, India

Mahesh Yambal[2]
Dept of computer Science
patel college of science &technology
Ratibadh,, Bhopal, India

Abstract-This paper compares the performance characteristics of two public key cryptosystems (RSA and ECC) used in digital signatures to determine the applicability of each in modern technological devices and protocols that use such signatures over the field *GF(p)* or *GF(2^m)* on an 50-MHz, 32-bit ARM microprocessor. Digital signatures are used in message transmission to verify the identity of the sender and to ensure that a message has not been modified after signing. The space and time efficiency of digital signature algorithms is essential to their widespread adoption in message transport systems.

Keyword-Public key cryptography, Digital Signature, GF, ARM Microprocessor, etc.

I. INTRODUCTION

The increasing use of network-connected embedded devices and online transactions creates a growing demand of network security for embedded systems. The security requirements, such as authentication, confidentiality and integrity, always make computationally intensive processes and can easily become the bottleneck of the related applications. Digital signatures are used in message transmission to verify the identity of the sender of the message and to ensure that the message has not been modified after signing. They are essential for verifying the authenticity of a message. The application of digital signatures is widespread in digital computing, taking the place of an ordinary hand-written signature. Because digital signatures are akin to hand-written signatures, they are used in many of the applications of signatures on the Internet (e.g. e-voting, online banking, online college applications, etc).

The risk of intrusion and eavesdropping goes up as electronic communication equipment becomes increasingly wireless and ubiquitous. With the spectre of hackers/crackers looming, security is becoming a major consideration in a growing number of embedded systems. Although sometimes mandated through standards, security is often added as feature to make a well-designed system more competitive in the marketplace.

While the benefits of strong, efficient security may be obvious, implementing it is not always straightforward and can be daunting when the security must function in extraordinary environments where battery life and processing power are low. The importance of digital signatures in digital communications merits the research into relatively new cryptosystems such as elliptic curve cryptography (ECC), especially as the need for more efficient algorithms grows with the growing number of memory-limited mobile electronic devices. The increasing key sizes needed by RSA for security against brute force attacks by powerful computers or distributed computing also makes ECC more appealing, with its smaller secure key sizes.

The integration of the public-key cryptographic techniques is often delayed or completely ruled out due to the difficulty of obtaining efficient, reliable solutions. It is obvious that we need

• Public-key cryptographic systems with higher strength per key bit.

• Efficient platform, specific and optimized implementations for a given restricted environment.

We analyzed the major bottlenecks at function level in ARM processors (90% of mobile phones use it) and evaluated the performance improvement, when we introduce some simple architectural support in the ARM ISA. We also address the problem of the power requirements of the benchmark; DPA techniques are a serious threat for the security of embedded devices and power analysis could be useful to achieve a more secure execution.

II. PUBLIC KEY CRYPTOGRAPHY

The data transferred from one system to another over public network can be rotected by the method of encryption. On encryption the data is encrypted/scrambled by any encryption algorithm using the ‘key’. Only the user having the access to the same ‘key’ can decrypt/de-scramble the encrypted data. This method is known as private key or symmetric key cryptography. But the problem with these algorithms is the key exchange. The communicating parties require a shared secret, ‘key’, to be exchanged between them to

have a secured communication. So, using public key algorithm a shared secret can be established online between communicating parties without the need for exchanging any secret data.

Public-key refers to a cryptographic mechanism. It has been named public-key to differentiate it from the traditional and more intuitive cryptographic mechanism known as: symmetric-key, shared secret, secret-key and also called private-key. Symmetric-key cryptography is a mechanism by which the same key is used for both encrypting and decrypting; it is more intuitive because of its similarity with what you expect to use for locking and unlocking a door: the same key. This characteristic requires sophisticated mechanisms to securely distribute the secret-key to both parties. Public-key on the other hand, introduces another concept involving key pairs: one for encrypting, the other for decrypting. This concept, as you will see below, is very clever and attractive, and provides a great deal of advantages over symmetric-key:

- Simplified key distribution
- Digital Signature
- Long-term encryption

In public key cryptography each user or the device taking part in the communication have a pair of keys, a public key and a private key, and a set of operations associated with the keys to do the cryptographic operations. Only the particular user/device knows the private key whereas the public key is distributed to all users/devices taking part in the communication. Since the knowledge of public key does not compromise the security of the algorithms, it can be easily exchanged online. A shared secret can be established between two communicating parties online by exchanging only public keys and public constants if any. Any third party, who has access only to the exchanged public information, will not be able to calculate the shared secret unless it has access to the private key of any of the communicating parties.

A. RSA Cryptography:

RSA is one of the oldest and most widely used public key cryptography algorithms. The algorithm was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman.

The RSA cryptosystem is based on the assumption that factoring is a computationally hard task. This means that given sufficient computational resources and time, an adversary should not be able to “break” RSA (obtain a private key) by factoring. This does not mean that factoring is the only way to “break” RSA (in fact, breaking RSA may be easier than factoring), but currently no other methods have been proposed to efficiently break RSA.

B. Elliptical Curve Cryptography:

As we said, public-key schemes are often used to set up private (symmetric) keys for encryption. Because hackers will attack the weakest link, it's necessary to match the strength of the private key used with that of the public, or asymmetric, key. ECC is a public-key system that's increasingly being used by organizations. It uses smaller key lengths than those used by older schemes (RSA, DSA) to achieve the same given security level. ECC is widely standardized--by National Institute of Standards and Technology (NIST), Federal Information Processing Standard (FIPS), and American National Standards Institute (ANSI).

Since its proposal by Victor Miller and Neal Kolbitz in the mid 1980s, Elliptic Curve Cryptography (ECC) has evolved into a mature public-key cryptosystem. Extensive research has been done on the underlying math, its security strength and efficient implementations. Small key sizes and computational efficiency of both public and private key operations make ECC not only applicable for hosts running on powerful machines but also to small wireless devices such as PDA’s and cell phones. To make ECC commercially viable, ECC protocols need to be standardized and integrated seamlessly into the security layers like SSL.

Every public key cryptosystem is based on some hard to solve mathematical property. By hard to solve we mean that even on the fastest of computers available today it is infeasible in terms of money and computational time to solve the problem. The popular RSA and Deffie-Hellman are based on the hardness of integer factorization as Discrete Logarithm Problem. Unlike these cryptosystems Elliptic Curve Cryptography operates on points on a Elliptic Curve. The basic operation in ECC is the point multiplication i.e. multiplication of an elliptic curve point P by an integer e, which we will denote by $e*P$.

III. DIGITAL SIGNATURE

A digital signature is an electronic analogue of a written signature; the digital signature can be used to provide assurance that the claimed signatory signed the information. In addition, a digital signature may be used to detect whether or not the information was modified after it was signed (i.e., to detect the integrity of the signed data). These assurances may be obtained whether the data was received in a transmission or retrieved from storage. A properly implemented digital signature algorithm that meets the requirements of this Standard can provide these services.

A digital signature algorithm includes a signature generation process and a signature verification process. A signatory uses the generation process to generate a digital

signature on data; a verifier uses the verification process to verify the authenticity of the signature. Each signatory has a public and private key and is the owner of that key pair. As shown in Figure 1, the private key is used in the signature generation process. The key pair owner is the only entity that is authorized to use the private key to generate digital signatures. In order to prevent other entities from claiming to be the key pair owner and using the private key to generate fraudulent signatures, the private key must remain secret.

Digital signature schemes can be used to provide the following basic cryptographic services: data integrity (the assurance that data has not been altered by unauthorized or unknown means), data origin authentication (the assurance that the source of data is as claimed), and non-repudiation (the assurance that an entity cannot deny previous actions or commitments). Digital signature schemes are commonly used as primitives in cryptographic protocols that provide other services including entity authentication, authenticated key transport, and authenticated key agreement.

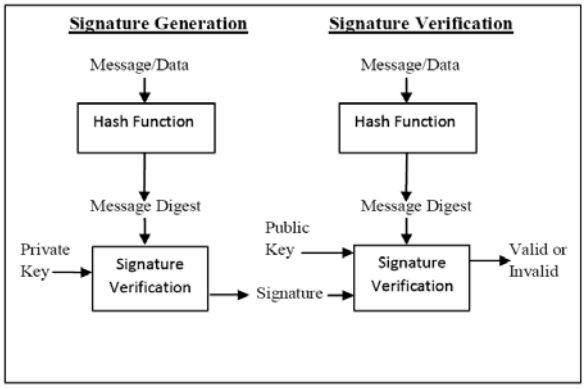

Figure 1. Digital Signature Process.

Figure 1 illustrates the basic working of Digital Signature Process from Signature Generation to Verification. The public key is used in the signature verification process (see Figure). The public key need not be kept secret, but its integrity must be maintained. Anyone can verify a correctly signed message using the public key. For both the signature generation and verification processes, the message (i.e., the signed data) is converted to a fixed-length representation of the message by means of an approved hash function. Both the original message and the digital signature are made available to a verifier.

IV. PROPOSED METHODOLOGY

The related works of the implementation of an efficient cryptographic algorithm for constrained devices have been focus on the optimizations of the multiplication operation, since it is the most expensive operation in RSA and ECC algorithms. There several implementations of RSA and ECC, ECC can be over either $GF(2^m)$ or $GF(p)$, we focus on modular arithmetic since this can be applied for RSA and ECC, and this has similar nature to the arithmetic of the micro-controller.

The majority of publications concentrate on finite field architectures for relatively small fields suitable for the implementation. Multiplication in $GF(2^m)$ is usually considered the crucial operation which determines the speed or throughput of a crypto system. Finite field architectures can be classified into bit serial (one output bit per clock cycle) and bit parallel ones (all output bits are computed within one clock cycle).

In the test of the asymmetric algorithms, RSA was compared to ECC. Due to the large amount of memory required by the asymmetric cryptography there was insufficient memory left to implement a PRNG, which is required to perform encrypt and signature-operations. Therefore the implemented operations were limited to decryption and verification. The two different algorithms were benchmarked with the previously determined key sizes. ECC was also implemented with the two different mathematical libraries, LibTomMath and TomsFastMath. The TomsFastMath library could not be used in combination with RSA due to lack of memory.

A. RSA Digital Signature

In the RSA digital signature process, the private key is used to encrypt only the message digest. The encrypted message digest becomes the digital signature and is attached to the original data.

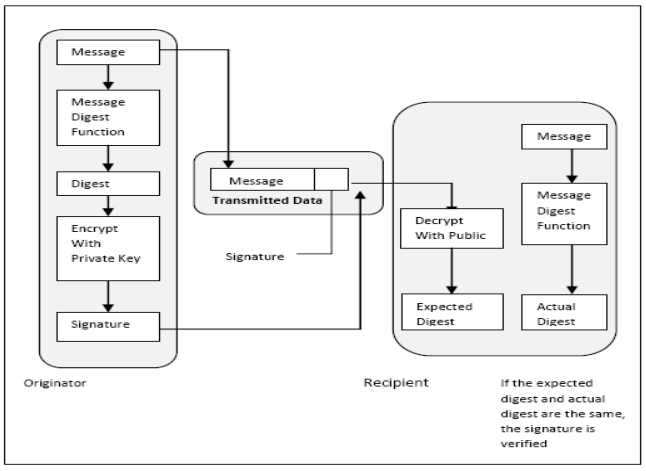

Figure 2. Basic RSA Data Security Digital Signature Process.

Figure 2 illustrates the basic RSA Data Security digital signature process. An RSA digital signature key pair consists of an RSA private key, which is used to compute a digital signature, and an RSA public key, which is used to verify a digital signature. An RSA key pair used for digital signatures **shall** only be used for one digital signature scheme. In addition, an RSA digital signature key pair **shall not** be used for other purposes (e.g., key establishment).

An RSA public key consists of a modulus *n*, which is the product of two positive prime integers *p* and *q* (i.e., *n* = *pq*), and a public key exponent *e*. Thus, the RSA public key is the pair of values (*n*, *e*) and is used to verify digital signatures. The size of an RSA key pair is commonly considered to be the length of the modulus *n* in bits (*nlen*).

The corresponding RSA private key consists of the same modulus *n* and a private key exponent *d* that depends on *n* and the public key exponent *e*. Thus, the RSA private key is the pair of values (*n*, *d*) and is used to generate digital signatures. Note that an alternative method for representing (*n*, *d*) using the Chinese Remainder Theorem (CRT) is allowed.

B. EC Digital Signature

The Elliptic Curve Digital Signature Algorithm (ECDSA) was first proposed in 1992 by Scott Vanstone in response to a National Institute of Standards and Technology (NIST) request for public comments on their first proposal for the Digital Signature Standard (DSS).

The ECDSA consists of separate tests for each of four distinct components. The various components that are tested with the ECDSA are:

1) Generation of Private and Public Key Pairs,
2) Public Key Validation,
3) Signature Generation, and
4) Signature Verification.

These ECC-based signatures are smaller and faster to create that aging RSA-based algorithms. The public key that the certificate holds is smaller and more agile as well. Verification is also faster using ECC-based certificates, especially at higher key strengths. For this reason, ECDSA is included as part of the NSA Suite B standard, helps secure the postal system, has been adopted by key standards, and used by leading vendors.

V. CONCLUSION

Different public key cryptosystems with different key sizes have different performance characteristics in Embedded Systems. In the implementation part, we will employ analytical techniques to analyze the performance of public-key cryptosystem (RSA and ECC) with Digital Signature operations on ARM7TDMI Microcontroller. The aim of this paper was to answer the question of whether it is feasible to implement advanced cryptography on an embedded platform to be used in an access system and if the resulting performance would be usable. So, from different references it is clearer that, ECDSA is having more secure strength than RSA Digital Signature.

REFERENCES

[1] Jonathan Lutz, Anwarul Hasan *High Performance FPGA based Elliptic Curve Cryptographic Co-Processor* Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04).

[2] Sandro Bartolini, Paolo Bennati, Roberto Giorgi, Enrico Martinelli *Elliptic Curve Cryptography support for ARM based Embedded systems.*

[3] Vivek Kapoor, Vivek Sonny Abraham, Ramesh Singh *Elliptic Curve Cryptography* ACM Ubiquity, Volume 9, Issue 20 May 20 – 26, 2008.

[4] Nicholas Jansma, Brandon Arrendondo *Performance Comparison of Elliptic Curve and RSA Digital Signatures* April 28, 2004.

[5] Ray C.C. Cheung, Wayne Luk, Peter Y.K. Cheung *Reconfigurable Elliptic Curve Cryptosystems on a Chip* Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05).

[6] R.L. Rivest, A. Shamir, and L. Adleman *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.*

[7] Albert Levi, Erkay Savas *Performance Evaluation of Public-Key Cryptosystem Operations in WTLS Protocol.*

[8] Govind Singh Tanwar, Ganesh Singh, Vishal Gaur *SECURED ENCRYPTION - Concept and Challenge* International Journal of Computer Applications (0975 – 8887) Volume 2 – No.3, May 2010.

[9] Lejla Batina, Nele Mentens, Bart Preneel, Ingrid Verbauwhede *Flexible Hardware Architectures For Curve-based Cryptography* Katholieke Universiteit Leuven, ESAT/SCD-COSIC, Belgium.