Invited Paper

Toward Faster Cryptographic Algorithms

Khaled Suwais
Faculty of Computer Studies, Arab Open University (AOU), Riyadh, Saudi Arabia
khaled.suwais@arabou.edu.sa

Abstract— many cryptographic algorithms are used to provide the cyber with various security services. The structures of these algorithms are found sequential in term of their data processing mechanisms. Therefore, current cryptographic algorithms are not prepared to embrace the enhancement brought by the multi-core processors. In this paper we propose few possible alternatives to bridge the gap between the powerful multi-core processors and the sequential structures of current cryptography.

Index Terms— Cryptography, Multi-core, Parallelism, Security.

I. INTRODUCTION

COMPUTER security plays an important role in securing our most important strategic resources – the information. The computer security can be viewed from three interconnected perspectives: system security, access control security and transmission security. The security level of any system is evaluated by verifying the security policy implemented in that system. The security policy is mainly designed to classify sensitive information, allow or forbid remote access to the system, define the trust level of users accessing the system from inside or outside the organization. From the other perspective, we need a secure access to the system control, by providing access authorization for specific users, controlling users access to system resources and prevent users from accessing private data of other users. From the transmission perspective, security must be responsible of protecting transmission from interception, use appropriate cryptographic primitives and encryption algorithms, and guarantee a high level of privacy for users during the transmission.

One of the best ways to secure the computer systems is through cryptography. Cryptography focuses mainly on issues of securing messages and providing authentication services, so that relevant parties can read the messages and verify the identity of the sender [1]. In fact, cryptography is an important part in computer security, which enables computer systems to ensure the information confidentiality, integrity, authentication and non-repudiation [2].

From the security point of view, cryptographic algorithms are expected to deliver the highest level of security services for computer systems. At the same time, these algorithms must be efficient in term of performance to comply with the intensive and sensitive transactions. The first part of this paper aims to introduce an overview study on current state of cryptographic algorithms in term of their structure and design. The main agenda of this paper is to examine the efficiency of the structure of current cryptographic algorithms in utilizing the new generation of the powerful multi-core processors. In addition, parallelism is introduced in this paper as a primary requirement for optimizing the utilization of multi-core processors.

The rest of the paper is organized as follows: Section II surveys the current state of sequential cryptography. In Section III we introduce parallelism and multi-core processors as essential keys for parallel cryptographic algorithms. Section IV presents possible directions in parallel cryptography. Lastly, the conclusion of our research paper is presented in Section V.

II. CURRENT STATE OF SEQUENTIAL CRYPTOGRAPHY

There are several cryptographic algorithms that are designed to deliver various cryptographic services for both on-line and off-line computer transactions. These cryptographic algorithms are generally divided into three categories: symmetric key (secret key) algorithms, asymmetric key (public-key) algorithms and un-keyed algorithms as shown in Fig. 1. Each one of these categories includes other algorithms of similar properties.

![Fig. 1. Classification of cryptographic algorithms](image)

In this section we study the structure of the categories shown in Fig. 1, in order to understand the nature of their data
processing mechanisms.

A. Analyzing Symmetric Key Algorithms

There are two types of symmetric key algorithms: stream ciphers and block ciphers. Both of stream ciphers and block ciphers use single input key to generate multiple keystream during data encryption. The difference between the two types is found in the encryption mechanism. Stream cipher uses single byte as a unique keystream (or bit in some stream ciphers) to encrypt single byte of plaintext (or bit), while block cipher performs the encryption only on a fixed-length group of bits (for both plaintext and keystream) called block.

Block Ciphers

Block and stream ciphers are classified into sub-categories, based on their internal structure. In the first classification, block ciphers are grouped in two categories: Feistel-based and Non-Feistel-based block ciphers. Both Feistel and Non-Feistel-based block ciphers generate ciphertext based on sequential rounds as in TwoFish (Feistel-based) [3] and AES (Non-Feistel-based) [4]. Generally, block ciphers use the concept of \( n \) interconnected sequential rounds (where \( n \) is the number of rounds in a given block cipher) to generate a fixed-length block of bits, called the ciphertext.

Stream Ciphers

Stream ciphers are grouped in three categories: hardware-based, software-based and hybrid-based stream ciphers. In the first category, the keystream generator of the stream cipher relies on shift-registers to generate keystream. Examples on stream ciphers from this category includes the summation generator [5] and SNOW stream cipher [6], which are mainly based on the use of Linear-Feedback Shift Registers (LFSR) as a core of their keystream generator. The structures of stream ciphers which belong to this category are sequential in nature. Generating new keystream requires a sequential execution for \( m \) LFSRs (or any other register). From other perspective, some hardware-based stream ciphers use shift registers with other non-linear or Boolean function in order to achieve better randomness. This group of stream ciphers (e.g. LILI-128 [7]) also includes sequential structures, where the keystream generator requires an ordered execution between the shift-register and the non-linear function to generate new keystream.

The second category of stream ciphers is called the software-based stream ciphers. This category includes numerous stream ciphers which do not rely on any specific hardware in their implementations. Examples on this category include: T-function-based stream ciphers (e.g. TSC-1) [8], S-Box-based stream ciphers (e.g. MUGI) [9] and Logic-and-Arithmetic-based stream ciphers (e.g. RC4) [10]. Software-based stream ciphers are generally producing new keystream in sequential manner. For example, RC4 runs two algorithms (key schedule algorithm (KSA) and pseudo-random generator algorithm (PRGA)) sequentially, where KSA prepares the internal secret state of the input key, and PRGA uses this state to produce the output keystream using array permutation techniques.

The last category of our classification includes hybrid-based stream ciphers. The implementation of stream ciphers in the category relies on both of the hardware devices and software techniques. There are numerous stream ciphers in this category, such as ABC [11] and Polar Bear [12] stream ciphers. The implementation of ABC and Polar Bear stream ciphers depends on a sequential linking between shift-registers and multiple functions. For instance, ABC stream cipher uses 38 registers along with three functions (linear transformation, T-function and non-linear mapping function) to produce single keystream, while Polar Bear uses two LFSRs along with a dynamic permutation as a core at the keystream generator.

In summary, the majority of symmetric key algorithms appeared in the literature are designed to run sequentially. The sequential part of these algorithms comes from the keystream generators, which mostly require specific orders to execute dependent inter-components.

B. Analyzing Asymmetric Key Algorithms

Asymmetric key algorithms are classified into three categories: Encryption, Key Sharing and Digital Signature algorithms. The three categories differ from each other in term of the usage purpose of the algorithms. However, at the same time, all asymmetric key algorithms share the same properties of using a pair of keys in their operations. In this section we focus on analyzing each category from the algorithms structure perspective.

Encryption algorithms

In order to achieve the security on transmitted messages (or data stored on hard drive) using any asymmetric key encryption algorithm, the source is required to encrypt the message by using the destination’s public key. The receiver, in turn, is required to decrypt the message by using his private key. For that purpose, several practical asymmetric encryption algorithms appeared in the literature, which are based on hard mathematical problems for higher security level.

There are two important groups of encryption algorithm in this category: Number theory based and Elliptic curve based encryption algorithms. Example on number theory based algorithm is the RSA encryption algorithm [13]. This cryptosystem relies on the integer factorization problem, which requires performing exponential and modulo operations. On the other hand, Elliptic Curve Cryptosystem (ECC) [14] is based on the elliptic curve discrete logarithm problem, which requires a multiplication of multiple points on a selected elliptical curve over a finite field. Both ECC and RSA cryptosystems are running in sequential fashion. ECC multiply two points on elliptic curve to generate single secret key at one time, while RSA requires an execution of two dependent operations (exponential and modulo operations) in order to generate the secret key.

It is noticeable that both of the cryptosystems (number theory based and elliptic curve based) generate single secret key at a time, which refers to the absence of parallelism in the architecture of asymmetric key encryption cryptosystems.
**Key Sharing algorithms**

This is yet another category of asymmetric key algorithms, where the algorithms are designed to exchange session keys (secret keys) between users securely. Diffie-Hellman [15] and ECDH [16] are two popular algorithms used for exchanging keys securely. Similar to the asymmetric key algorithms discussed above, Diffie-Hellman and ECDH are based on discrete logarithm and elliptic curve cryptography respectively. The structure of key sharing algorithms can be viewed from the communication perspective and from on-side computation perspective. Both of Alice and Bob perform their (on-side) computation in parallel fashion, where Alice and Bob compute their secret values independently. From the other perspective, calculating the secret key is dependent process where each side depends on the exchanged secret value form the other side, which is a serial process.

However, current key sharing algorithms are able to exchange single secret key at one time. The parallel part of the key sharing algorithms structure, is not affecting the efficiency of the algorithms due to the existence of dependency on both sides during the on-side computations.

**Digital Signature algorithms**

Digital signature is a verification method which aims to provide authenticity service [17]. The private key is used to sign a message, while the public key is used to verify the message through the signature.

The Digital Signature Algorithm (DSA) [18] and Elliptic Curve Digital Signature Algorithm (ECDSA) [19] are two examples algorithm that produce digital signatures. Producing digital signatures in these two algorithms requires executing a set of sequential operations, where the internal operation execution in specific order is essential for accurate digital signatures.

**C. Analyzing Un-Keyed Algorithms**

Un-keyed cryptographic algorithms refers to algorithms that do not rely on specific key to work (refer to Fig. 1). Examples of such algorithms include Hashing and Message Digesting (MD) algorithms. These algorithms are developed to produce a fixed-length hash value (also known as message digest) of a given message. The produced hash value can then be used for authentication or other security purposes. The mechanism of producing a hash value of given messages.

There are many hash function algorithms such as SHA family [20] and MD5 algorithm [21]. The general structure of these algorithms relies on the use of compression functions to process message segments. The compression function of most hash function depends on sequential additive, shift and permutation operations over each message segment. Therefore, the dependency between the internal components of the hash algorithms forces the algorithm to produce message digests in sequential.

**III. PARALLELISM ON MULTI-CORE PROCESSOR**

Parallel processing is the concurrent use of multiple computer resources (e.g. multi-core, multi-processors, etc.) in order to solve computational problems. The given problem is broken into smaller segments, and then processed concurrently on the processor. Technically, achieving higher performance depends on the hardware devices and the processing techniques. In fact, these two factors measure and control the efficiency gained by parallelizing particular system on chosen computer architecture.

In present time, multi-core processors and multithreading techniques form the perfect combination for efficient parallelism. Multi-core processor provides systems with greater performance by providing extra cores in single physical processor. On the other hand, multithreading techniques work in lower levels, by allowing multiple threads to be associated with each core to accomplish one task. The expected performance obtained when employing multithreading on multi-core processor is expressed in Fig. 2.

![Fig. 2. Performance gained from multi-core processors (22)](image2)

Increasing the number of cores in multi-core processors showed better performance compared to adding multiple single-core processors. Performance analysis on multi-core processors conducted by Intel [23] shows that one Quad-core processor performs 66% faster than eight single-core processor, and performs 33% faster than one Dual-core processor as shown in Fig. 3.

![Fig. 3. Performance gained from multi-core processors [23]](image3)
paper we aim to bridge the gap between the sequential cryptographic systems and current computer processors, through the utilization of multi-core processors and parallelism techniques.

IV. FUTURE DIRECTION: PARALLEL CRYPTOGRAPHY

Parallel cryptographic systems have been studied from two perspectives: hardware-based parallelism and software-based parallelism. Hardware-based parallelism focuses on designing special parallelizable cryptographic hardware. Examples on hardware-based parallelism include: the implementation of Rijndael algorithm on VHDL (VHSIC Hardware Description Language) [24], the Field-Programmable Gate Array (FPGA) implementation of RSA [25], and the FPGA-based symmetric encryption algorithms [26].

On the other hand, software-based parallelism implements parallelism on two levels: job-level and instruction level. Parallelizing cryptographic systems was presented in several researches, such as parallelizing the three-layer model for elliptic curve scalar multiplication, which can be applied on different cryptographic applications due to its high performance.

However, in a bigger scale, changing the direction of sequential cryptographic algorithms to work in parallel is possible in three approaches: code parallelism, internal-design parallelism, and parallel architecture. In the first approach (code parallelism), parallel constructs (e.g. OpenMP pragmas) are applied to the code of the algorithm. These constructs will automatically handle parallelizing the entire code of the algorithm, resulting in higher performance. This approach is relatively easy to implement, but at the same time, the approach can only achieve minimal level of concurrency.

The internal-design parallelism approach requires parallelism at the design phase of new cryptographic algorithms. This approach will highly parallelize new algorithms, but it is considered hard and time consuming process.

Both of the code parallelism and internal-design parallelism approaches requires changes on every cryptography algorithms. On the contrast, designing a parallel architecture (third approach) is suitable in accommodating numbers of cryptographic algorithms. The parallel architecture is designed such that the existing algorithms can run in parallel fashion. However, this approach is suitable for only selective algorithms of similar structures, and therefore special concern must be given during the designing phase of the parallel architecture components.

Adopting any parallelism approach for parallelizing cryptographic algorithms must be done properly to utilize multi-core processors, since efficient parallel implementation requires optimal utilization of multi-core processors. The chosen parallel approach must be scalable in term of its dynamic adoption of the increasing number of cores.

The architecture of scalable parallel approach must include Intelligent Core Detector (ICD), able to detect the number of cores on any computer system. Consequently, ICD communicates with Thread Generation Engine (TGE) to generate suitable number of threads, based on the detected number of cores as shown in Fig. 4.

![Fig. 4. Essential elements for parallel cryptographic architecture](image)

In the process of parallelizing cryptographic algorithms based on the parallel architecture (refer to Fig. 4), designer should consider two important issues for higher efficiency: memory management and workload distribution. The architecture is designed such that the workload of the algorithm is distributed equally among the cores (achieved by the Workload Distributer Component). At the same time, the architecture needs to determine the optimum amount of data to be processed at one time, such that it minimizes the memory accessing time (achieved by the Memory Manager Component).

Parallelizing cryptographic algorithm (from the programming perspective) can be achieved by multithreading techniques. The thread engine (TGE) generates a set of threads to be associated with the algorithm parallelizable components. The generated threads along with the workload distributor enable the parallel architecture to maintain optimum level of stability and workload balancing. The overall structure of the parallel architecture is portrayed in Fig. 5.

![Fig. 5. Internal structure of the parallel architecture](image)

V. CONCLUSION

This paper had identified a gap between the implementation of sequential cryptographic algorithms and the future of multi-
core processors. Subsequently, three possible alternatives are given as suggestions in view of helping bridging the gap. These alternatives are designed with a focus on optimum utilization of the multi-core processors. It is expected that by exploring the proposed alternatives, new essential findings will be identified to benefit the development of future parallel cryptography.

VI. REFERENCES