computers

MDPI

Article

Mixed Cryptography Constrained Optimization for Heterogeneous, Multicore, and Distributed Embedded Systems

Hyunsuk Nam and Roman Lysecky *

Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA; hnam@email.arizona.edu
* Correspondence: rlysecky@ece.arizona.edu; Tel.: +1-520-621-6192

Received: 28 February 2018; Accepted: 22 April 2018; Published: 24 April 2018

Abstract: Embedded systems continue to execute computational- and memory-intensive applications with vast data sets, dynamic workloads, and dynamic execution characteristics. Adaptive distributed and heterogeneous embedded systems are increasingly critical in supporting dynamic execution requirements. With pervasive network access within these systems, security is a critical design concern that must be considered and optimized within such dynamically adaptive systems. This paper presents a modeling and optimization framework for distributed, heterogeneous embedded systems. A dataflow-based modeling framework for adaptive streaming applications integrates models for computational latency, mixed cryptographic implementations for inter-task and intra-task communication, security levels, communication latency, and power consumption. For the security model, we present a level-based modeling of cryptographic algorithms using mixed cryptographic implementations. This level-based security model enables the development of an efficient, multi-objective genetic optimization algorithm to optimize security and energy consumption subject to current application requirements and security policy constraints. The presented methodology is evaluated using a video-based object detection and tracking application and several synthetic benchmarks representing various application types and dynamic execution characteristics. Experimental results demonstrate the benefits of a mixed cryptographic algorithm security model compared to using a single, fixed cryptographic algorithm. Results also highlight how security policy constraints can yield increased security strength and cryptographic diversity for the same energy constraint.

Keywords: security-driven optimization; heterogeneous multicore systems; mixed cryptographic security model; adaptive system; runtime security optimization; system-level codesign; distributed systems

1. Introduction

Distributed, heterogeneous embedded systems are spreading widely in numerous applications, including video-based object detection and tracking [1], automotive systems, automated greenhouses [2], and Internet of Things, among others. Distributed embedded systems are composed of numerous embedded devices incorporating various sensors, actuators, and heterogeneous computing resources. Those heterogeneous computing resources include processors, which may vary by the type and number of cores, application-specific hardware accelerators, reconfigurable computing resources such as field-programmable gate arrays (FPGAs), GPUs, etc. Distributed embedded systems may also communicate with servers to offload computationally-intensive operations or store and retrieve data. Depending on the application domain, such communication may use wired or wireless networks. Computing resources, both local and distributed, have performance and energy constraints that must be considered in mapping and optimizing an application onto these distributed heterogeneous

architectures. Many applications are dynamic with runtime changes in data inputs, operational modes, and system constraints. As data and system constraints change, the underlying algorithms and system performance requirements may change, which in turn requires re-optimizing the system to achieve the best performance for current needs.

Two of the most critical design concerns for distributed embedded systems are energy and security. Data confidentiality plays an important role in communication between tasks both between devices as well as within devices. To increase security, cryptographic algorithms can be used when communicating data within or between embedded devices, but such security comes with a tradeoff of increased energy consumption and latency. As mobile or battery-powered embedded devices have limited energy availability, design methods and tools for adaptive, distributed, and heterogeneous systems should consider security, latency, and energy tradeoffs in an integrated approach when mapping and optimizing an application onto these platforms.

Security-driven optimization of embedded systems can be categorized as security-integrated, security-constrained, and security-optimized. Security-integrated approaches incorporate security within the system design, such as using a specific cryptographic algorithm for all inter-device communication [3], but does not use security as a constraint or optimization metric. Security-constrained approaches define system constraints that are directly related to security, such as requiring a minimum number of rounds with the Rijndael encryption algorithm [4], but do not attempt to optimize security. In contrast, a security-optimized approach uses security metrics within the system fitness evaluation with a goal of optimizing security, either as the primary objective or as part of a multi-objective optimization. For example, Zhang et al. [5] presented a security-optimized approach that optimized the cryptographic algorithm used to communicate between tasks within a real-time application implemented using a multicore system. However, this approach only considers software tasks executing on homogeneous multicore systems, but does not consider heterogeneous resources or distributed embedded systems.

In this paper, we present a security-constrained and security-optimized approach for optimizing distributed, heterogeneous embedded systems using mixed cryptographic implementations. We consider distributed embedded systems incorporating heterogeneous embedded devices, each of which may include different processors and/or FPGAs. The presented methodology supports wireless communication using cryptography for both inter-device and intra-device communication (i.e., all task-to-task communication can be encrypted to maintain confidentiality). The mixed cryptographic implementations include multiple symmetric and asymmetric algorithms, with different configurations of each algorithm to support varying security levels. Using a dataflow-based application modeling framework, which incorporates models for computational latency, cryptographic security levels, communication latency, and energy consumption, the proposed framework supports optimization of latency, energy, and/or security metrics across all computing and communication levels [6]. Enabled by the use of a mixed cryptography model, we consider the integration of constraints for implementing specific security policies, which are implemented as hard constraints within the optimization algorithm. We evaluate the presented methodology using a video-based object detection and tracking application, and three additional synthetic applications, representative of applications with differing computational and communication ratios. We further evaluate the energy and security tradeoffs from using a reduced, yet diverse, set of cryptographic implementations and from using several runtime security policy constraints.

2. Related Work

Numerous approaches have addressed the schedulability and optimization of real-time systems using distributed and heterogeneous embedded systems. For automotive systems, consisting of interconnected devices using wired communication, several approaches have focused on scheduling and mapping of tasks to devices (e.g., ECUs) [7,8]. However, these approaches do not consider hardware accelerators, cryptography for inter-device communication, or the use of both wired and

wireless communication networks. Instead of optimizing the mapping and scheduling of tasks for a fixed distributed architecture, Pomante et al. [9,10] presented a design space exploration approach that simultaneously explores the heterogeneous components (e.g., general-purpose processors and domain specific processors) integrated within the distributed embedded devices. For systems in which the targeted distributed architecture has not been fixed, exploring the definition of the embedded devices can enable better optimization. However, this approach does not consider cryptography for communication or heterogeneous communication.

Shang et al. [11] presented a hardware/software optimization method that optimizes both system cost and power consumption for low-power, real-time systems. This proposed method uses time multiplexed scheduling to reconfigure distributed FPGAs at runtime to execute sequential tasks, while minimizing the latency of the reconfiguration schedule and reducing the energy of reconfiguration. In contrast, this paper presents an optimization method that employs a dataflow-based application model in which hardware implemented in reconfigurable resources is not reconfigured dynamically, but rather is used in order to meet application requirements.

Previous efforts have resulted in security-aware optimization methods for embedded systems. Lin et al. [12,13] presented a security-aware methodology that incorporates authentication methods within time division multiple access (TDMA)-based real-time distributed embedded systems (e.g., FlexRay). Given security and latency constraints, this approach determines the task allocation, priority assignment, network scheduling, and key-release intervals. In addition, a security-constrained and security-optimized optimization method utilizes a path-based security constraint to minimize risk, where security risk is defined as the risk that two tasks use the same encryption key. However, this approach only considers homogeneous multicore systems, but does not consider heterogeneous resources, distributed systems, or mixed cryptographic implementations.

Gu et al. [4] considered a task mapping on FlexRay based distributed hardware platform to meet security and latency constraints for minimizing the number of hardware coprocessors needed in the system. This approach is a security-constrained approach because they focused on minimizing the total number of HW units needed for a given hardware platform of multiple ECUs connected by a FlexRay bus subject to security and latency constraints. While such approaches seek to optimize the cryptography and authentication methods used in distributed automotive electronics, inter-device cryptography, wireless communication, and energy constraints are not considered.

Jiang et al. [14] presented a security-constrained hardware/software optimization method for automotive systems composed of a fixed number of ECUs and a configurable number of FPGAs, communicating over a CAN or FlexRay bus. Given a designer-specified security requirement (i.e., minimum number of rounds in Rijndael encryption algorithm) and real-time latency constraints, the hardware/software optimization minimizes the number of FPGAs required to meet those requirements.

Several research efforts have also analyzed the latency, energy, and memory tradeoffs of different cryptographic algorithms and implementations [15]. Pous and Joancomarti [3] analyzed various symmetric and asymmetric cryptographic algorithms, hash chain functions, elliptic curve cryptography, etc., and compared them with the costs of basic operating system functions, thereby quantifying the overhead that a secure protocol introduces. Mansour and Chalhoub [16] evaluated different symmetric and asymmetric security algorithms within wireless sensor networks to evaluate the time, energy, and memory usage, concluding that asymmetric cryptography has a significant effect on energy consumption of sensor nodes. Notably, such efforts provide the foundation upon which to build estimation models that are needed within system-level design methods and optimization algorithms.

Peter et al. [17] considered the system-level design and optimization of a system-on-a-chip (SOC) incorporating hardware accelerators for AES and ECC cryptographic algorithms, while also considering the trade-off between software and hardware. This approach and architecture effectively determines a single optimized cryptographic implementation and a single hardware accelerator for all communication, whereas the approach presented herein supports different cryptographic

implementations for each pair of communicating tasks. Additionally, tasks implemented in hardware will incorporate dedicated components for the cryptographic algorithms used by that task to ensure high system throughput.

Other research has focused on exploring cryptosystems with mixed cryptographic implementations [18,19]. Kuppuswamy and Al-Khalidi [18] proposed a hybrid cryptosystem that combines the convenience of a public-key cryptosystem with the efficiency of symmetric key cryptography, demonstrating that their proposed cryptography algorithm provides increased security and authentication compared to other hybrid algorithms. This approach seeks to develop a hybrid cryptographic implementation that could be used for specific communication channels, whereas the mixed cryptographic approach presented in this paper uses different cryptographic algorithms for different communication channels. Even further, some research has addressed the design of secure processor architectures, including the use of secure coprocessors for cryptographic operations [6,20,21], and developing secure computing architectures that separate processing into secure and insecure components [22]. We note that these efforts are complementary to the approach presented herein.

3. Threat Model

The considered threat model assumes malware can affect both software and hardware components across embedded devices within the distributed embedded system. Since FPGAs can be dynamically reconfigured at runtime by software, malicious software can also reconfigure parts of an FPGA to implement malicious hardware circuitry, which enables an easier path to inserting hardware Trojans compared to ASIC-based implementations [23]. Hardware-based malware configured in an FPGA will have access to the system bus, which thereby enables the malware to potentially eavesdrop on all intra-device communication between tasks, including tasks implemented in both software and hardware. Additionally, with access to the system bus, malicious hardware can not only monitor communication between software and hardware tasks within the device, but also enables a covert communication channel between malicious system components [24]. Thus, malware threatens the confidentiality of all inter-device and intra-device communication, regardless of the resources type in which a task is implemented. To address this threat, we consider a security policy that utilizes cryptography to achieve confidentiality for all intra-device and inter-device communication between tasks.

4. Security-Driven Optimization Methodology

Figure 1 presents an overview of the proposed security-driven optimization for distributed heterogeneous embedded systems. To enable efficient optimization considering latency, energy, and security constraints, integrated models are needed for applications, embedded systems architectures, and security methods. Our approach utilizes three integrated models, namely a dataflow model for the application incorporating estimation methods for latency and power consumption of application tasks on heterogeneous software and hardware resources, an embedded system architecture model that specifies the processor and reconfigurable resources and the composition of those devices into the distributed architecture, and a security model for capturing the mixed symmetric and asymmetric cryptographic algorithms used to transmit data between all tasks. Within the security model, a simplified metric is defined to quantify the relative security levels of the mixed cryptographic algorithms. We refer the interested reader to [25] for further details of the estimation framework used to estimate the latency and power consumption of software, hardware, and communication. However, we note here that the estimation methods used within our approach are based on physical measurements of prototypical implementations.

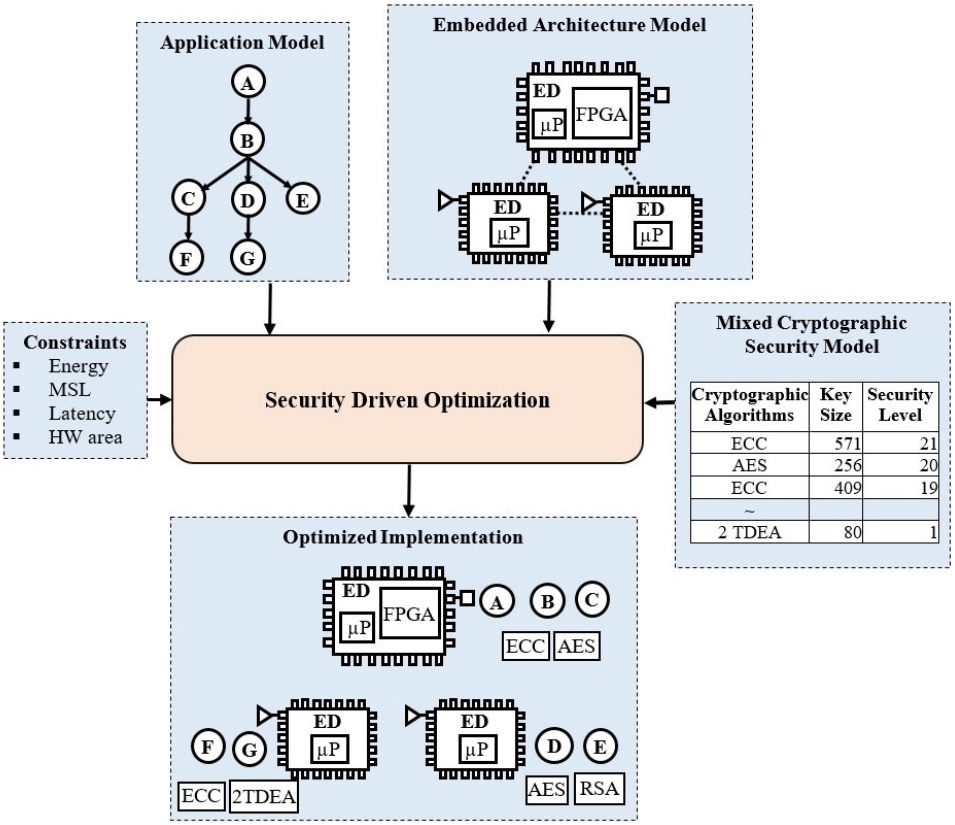


Figure 1. Overview of the methodology of the security-driven optimization.

4.1. Application Modeling

4.1.1. Dataflow Model

The application model uses a parameterized synchronous dataflow (PSDF) model [26] to specify system tasks and tokens transmitted between tasks. Each task T_i defines a specific computation/algorithm that fires when all required input tokens are available. TS_{ij} defines the size of tokens transmitted from task T_i to task T_j, which is dependent on system parameters.

Figure 2 presents the dataflow model for a video-based vehicle detection and tracking application, which we utilize to illustrate the application model. Given a video input, the first high-level operations extract the vertical and horizontal projections, which include the tasks' horizontal difference (HD), horizontal projection histogram (HP), vertical difference (VD), and vertical projection histogram (VP). The segmentation (SG) task determines the regions in the video for individual objects. The inverse wavelet transform (IWT) and support vector machine (SVM) are utilized to perform image classification to determine the identified object type. Finally, the autoregressive-moving average (AR) task processes the location of objects between frames to track object movement. The labels for each edge in Figure 2 present the token size in words for communication between tasks.

To support high throughput and performance, dataflow models often employ stream-based execution methods in which data is directly transmitted between tasks and stored locally to avoid memory contention that results from storing all data in global memory. Our approach employs such a stream-based processing approach, supported by the communication middleware, and further requires that only encrypted data is stored in global memory. This restriction is imposed to ensure that that malware cannot directly access unencrypted data in global memory.

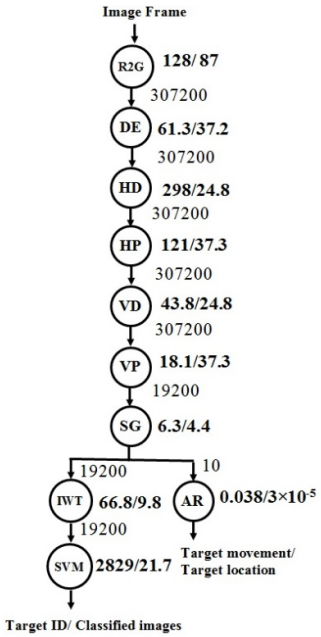


Figure 2. Overview of the application model for a video-based vehicle detection and tracking application. Labels to the right of each node indicate the latency for the task's software and hardware alternatives for a base device. Labels for edges indicate the size of tokens transmitted between tasks.

4.1.2. Execution Latency Model

The application model incorporates latency estimates for at least one software implementation and one hardware implementation, each defined for a base operating frequency. The labels for each task within Figure 2 present the base software and hardware latency for the video-based vehicle detection and tracking. For example, the label 128/87 for R2G indicates a latency of 128 ms for a software-based alternative and 87 ms for the hardware implementation. The latency estimate defined within the application model is defined for a specific embedded device (ED). The estimates in Figure 2 are defined for an embedded device with a 700 MHz ARM CortexA-15 processor and a reconfigurable FPGA with a maximum operating frequency of 100 MHz. Whereas the maximum frequency for software execution is dependent only on the processor's maximum frequency, the maximum operating frequency of hardware is dependent on both the target FPGA device and the hardware design itself, and must be determined by synthesizing the hardware task implementation for the target FPGA.

As different EDs may have different processors and operating frequencies, the execution time of a task executing on a different ED than the base latency specification must be estimated. Without loss of generality, we currently consider frequency scaling of software tasks across EDs, assuming the same processor architecture. For example, the HP task's software execution latency of 121 ms at 700 MHz becomes 70 ms for an ED with a frequency of 1.2 GHz.

4.1.3. Communication Latency

We utilize an efficient communication middleware that supports the parameterized dataflow model [26] and enables direct communication between all software and hardware task implementations. Inter-device communication is assumed to use wireless communication (i.e., IEEE 802.11 g). Since communication latency is tightly coupled to each embedded architecture, we utilized physical measurements from prototypical implementations to determine accurate estimates of communication latency for all possible communication modes and various token sizes. Given the communication latency measurements for the selected token sizes, linear or quadratic regression was used to determine the functions for estimating the communication latency as a function of the words transferred.

4.2. Mixed Cryptography Security Model

The security model integrates modeling of security levels for encryption/decryption of tokens transmitted between tasks and the latency of encryption/decryption based on the implementation (i.e., hardware vs. software). To provide diversity and robustness for securing inter- and intra-device communication, we utilize a mixed cryptographic implementation combining symmetric and asymmetric cryptography. Symmetric cryptography algorithms include Rijndael encryption [27,28] and TDEA [27,28], and asymmetric algorithms include RSA [29] and ECC [27]. For each implementation, we consider multiple key sizes, using the NIST recommendations [30]. In addition to key size, the Rijndael encryption algorithm also supports configuring the number of rounds, for which we consider a minimum of 10 rounds and a maximum of 14. In this paper, we consider ECC over binary fields due to the efficient hardware implementation thereof [31,32], but note that our approach can support other ECC implementations (e.g., ECC over prime fields [33]).

We employ an approach that models the strength of different encryption/decryption algorithms by ranking the algorithm alternatives, similar to criticality levels in real-time systems [6]. The security level defines a relative ranking of strength of the selected cryptography method. A higher security level provides stronger encryption/decryption compared to a lower security level. An alternative approach is to define the strength such a metric based on key size, such as using the equivalent Rijndael encryption key size. However, because cryptographic implementations may have different configurable options, providing an equivalent key size may not be possible for all configurations. Considering the Rijndael encryption algorithm itself, increasing the number of rounds can increase the strength, but does not affect the key size, and defining an increase in the equivalent key size for each additional round is impractical. The security level can capture the effects of various configurations and can be directly translated to a quantitative metric, thereby yielding a monotonically increasing function of cryptographic strength. Additionally, the security level enables the efficient encoding of cryptographic options with a genetic optimization algorithm (discussed in Section 4.5).

Table 1 presents the mixed cryptographic (MC) implementations considered in this paper, specifying the different key sizes (and rounds for Rijndael encryption) used for each algorithm, the security level for each implementation and, for reference, the equivalent Rijndael encryption key size [30]. We additionally consider a restricted mixed cryptography (MCR) model that only utilizes the maximum number of rounds for each key size within the MC model for Rijndael encryption, presented in Table 2.

Table 1. Security levels for the mixed cryptography (MC) security model.

Security Level	Cryptographic Algorithms	Key Size/Rounds	Equivalent Rijndael Key Size
21	ECC	571	285.5
20	Rijndael	256/14	256
19	Rijndael	256/13	256
18	Rijndael	256/12	256
17	Rijndael	256/11	256
16	Rijndael	256/10	256
15	ECC	409	204.5
14	Rijndael	192/13	192
13	Rijndael	192/12	192
12	Rijndael	192/11	192
11	Rijndael	192/10	192
10	ECC	283	142
9	Rijndael	128/12	128
8	Rijndael	128/11	128
7	Rijndael	128/10	128
6	ECC	233	116.5
5	RSA	2048	80
4	3 TDEA	112	80
3	ECC	163	81.5
2	RSA	1024	80
1	2 TDEA	80	80
0	None	0	0

Table 2. Security levels for the mixed cryptography restricted (MCR) security model.

Security Level	Cryptographic Algorithm	Key Size	Equivalent Rijndael Key Size
12	ECC	571	285.5
11	Rijndael	256	256
10	ECC	409	204.5
9	Rijndael	192	192
8	ECC	283	142
7	Rijndael	128	128
6	ECC	233	116.5
5	RSA	2048	112
4	3 TDEA	112	112
3	ECC	163	81.5
2	RSA	1024	80
1	2 TDEA	80	80
0	None	0	0

For each cryptographic algorithm, a software- and hardware-based implementation is available, the selection of which depends on a task's implementation and communication with other tasks. For software-based cryptography, a single software implementation of each algorithm is used by all software. For tasks implemented in hardware, we assume a dedicated encryption and decryption component is used for each communication channel.

In our current model, we focus specifically on the cryptographic implementation itself, and not on the evaluation/optimization of communication protocols (e.g., TLS, SSL) or key management, both of which are left as future work. Instead, this paper seeks to define a foundational framework for evaluating the use of cryptographic methods of differing strength for both intra-device and inter-device communication.

4.3. Embedded System Architecture Model

4.3.1. Embedded Device

Adaptive, distributed embedded systems can be defined as being composed of heterogeneous embedded devices EDs connected by wireless and/or wired connections, along with cloud computing resources for storing data or offloading computations. The heterogeneous EDs may have different computing resources, including processor types, number of processor cores, custom hardware accelerators, or reconfigurable FPGAs. Figure 3 presents an overview of the four distributed embedded devices considered in this paper to evaluate security-driven optimization.

The distributed heterogeneous architecture A1 consists of two EDs. ED1 incorporates an ARM processor operating at 700 MHz and an FPGA with a maximum frequency of 100 MHz and a system bus frequency of 100 MHz. The FPGA is divided into a maximum of four equal reconfigurable regions, and each hardware task implementation is constrained to the size of one reconfigurable region. In this architecture, ED1 serves as a source node, which is used for the targeted video-based vehicle detection and tracking application. ED2 incorporates a single processor operating at 1.2 GHz with a system bus frequency of 600 MHz, and serves as the sink node for the target application.

Architecture A2 consists of three EDs, including a sink ED3 incorporating a single ARM processor operating at 1.8 GHz with a system bus frequency of 1333 MHz and two source ED4s incorporating a single ARM processor operating at 1 GHz with a system bus frequency of 533 MHz.

Architecture A3 consists of three EDs, including ED5. ED5 incorporates the most capable multicore processor: a quad-core processor operating at 1.8 GHz with a system bus frequency of 1333 MHz. ED5 supports per-core shutdown that enables individual cores to be shut down if not utilized.

Lastly, architecture A4 includes ED6, which incorporates a MicroBlaze processor operating at 200 MHz, an FPGA with a maximum frequency of 150 MHz, and a system bus frequency of 150 MHz. Notably the FPGA within ED6 is more than 10X larger than the FPGA within ED1.

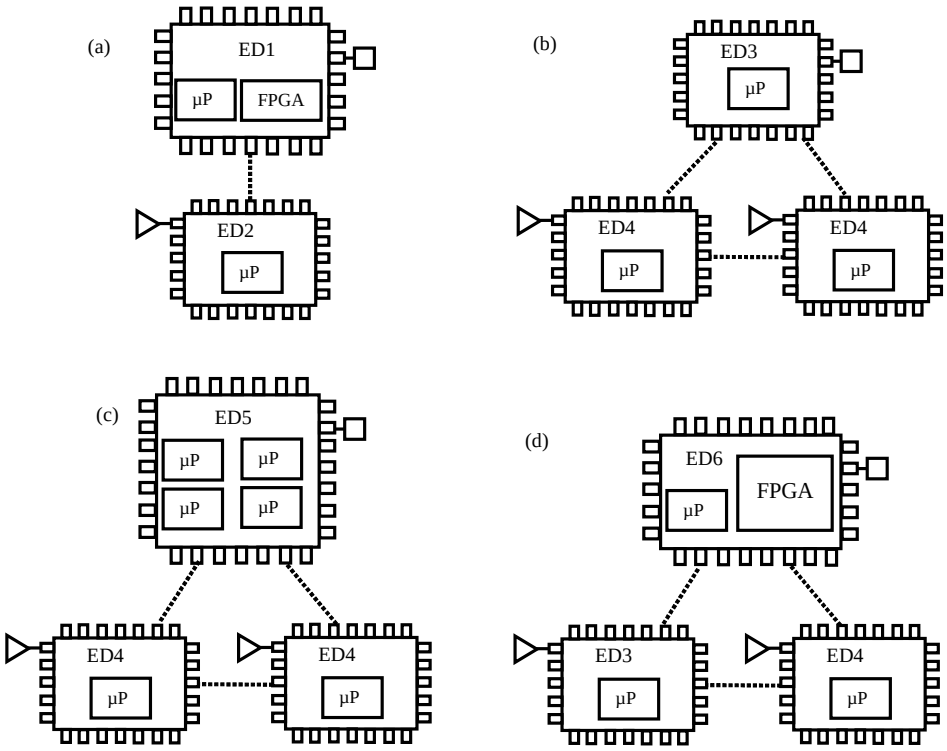


Figure 3. Overview of four distributed embedded architectures (**a**) A1 architecture; (**b**) A2 architecture; (**c**) A3 architecture; and (**d**) A4 architecture (A1–A4) comprised of several heterogeneous embedded devices (ED1–ED6).

4.3.2. Power Consumption

Power consumption is estimated as the total power consumed across all EDs in the distributed embedded system. The power consumption of each ED, P_{ED}, is modeled as four separate components, software power, P_{SW}, hardware power, P_{HW}, communication power P_C, and power consumption of cryptography for security, P_S.

The software power consumption is based upon the active and idle power consumption of the processor within the ED. The percentage of time the processor is idle or active is determined using the execution, communication, and security latency models for each task mapped to each processor core. Communication power is the total time spent communicating between tasks for all modes of communication. Given the communication latency for all software tasks, the software power estimates are utilized to calculate the communication power.

Since power consumption of hardware is specific to the hardware task implementation and FPGA, the active and idle power consumed for each hardware task implementation must be separately specified. Those power consumption estimates can come from numerous sources, including high-level synthesis, post-implementation simulation, physical measurement for the target FPGA prototype, etc.

Power consumption of software-based cryptographic implementations is determined by calculating the total latency combined with the software power consumption estimates. The power consumption of software-based encryption is a function of the security level utilized, the amount of data being encrypted, and the frequency of the processor. For hardware, the power consumption of each encryption and decryption component is separately calculated from a prototype hardware implementation of each cryptographic implementation.

4.4. Energy Optimization Methods

The security-driven optimization seeks to map application tasks to hardware or software on the distributed heterogeneous EDs and optimize the average security level across all tasks subject to constraints on latency, energy, and a minimum required security level. Within this optimization, we consider two energy optimizations methods, namely dynamic voltage and frequency scaling (DVFS) and per-core shutdown.

First, DVFS is utilized for the processor cores within all EDs considered. The processor cores' frequency ranges from a minimum of 100 MHz to a maximum of 1.8 GHz (depending on the ED). For each processor core, we have defined a finite set of operating points based on 100 MHz increments in the processor frequency. Additionally, while voltage scaling is supported within the FPGA in ED1 and ED6, frequency scaling is utilized with frequencies ranging from 10 MHz to 100 MHz, scalable in 10 MHz increments. Second, we consider per-core shutdown to reduce energy consumption on processor cores within ED5 when individual cores are not needed. We note that per-core shutdown is performed statically and not dynamically, meaning the decision to use cores is based on the static mapping of tasks to cores.

4.5. Genetic Optimization Algorithm

A multi-constraint, multi-objective genetic optimization algorithm was developed for the security-driven optimization. System constraints can include end-to-end latency, minimum security level, energy consumption, and resource constraints on the number tasks implemented in hardware within the FPGA. While the optimization methodology presented herein can support different constraints and optimization goals, we currently focus on simultaneously minimizing end-to-end latency and maximizing average task security level, given constraints on energy consumption and a minimum required security level.

To support the security-driven optimization, the genetic algorithm utilized three chromosomes, *M*, *C*, and *F* to encode the mapping of tasks to cores, the assignment of cryptographic implementations to communication channels, and the operating frequency and voltage for each task, respectively.

For task mapping, the chromosome represents the possible mapping of tasks to hardware or software alternatives within a specific ED. For the *M* chromosome encoding, tasks are ordered by depth first traversal of the application model. For the video-based vehicle detection and tracking application, the task ordering is: R2G, DE, HD, HP, VD, VP, SG, IWT, SVM, and AR. To encode the mapping of tasks to specific implementations, a unique numerical ID is used for each unique implementation option. For example, for architecture A1, 0 corresponds to software implementation on ED1, 1 correspond to hardware implementation on ED1, and 2 corresponds to software implementation on ED2. For example, the chromosome (0,0,0,0,0,0,2,1,1,2) encodes the mapping of the video-based vehicle detection and tracking application in which the R2G, DE, HD, HP, VD, and VP tasks are mapped to software on ED1, the tasks SG and AR tasks are mapped to hardware on ED1, and the IWT and SVM tasks are mapped to software on ED2.

For the cryptographic implementation, as the security level is unique to each cryptographic option, the *C* chromosome uses the security level to encode the cryptographic implementation used for each pair of communicating tasks. Again, a depth-first traversal of the application model is used to order the communication channels within the chromosome. Considering the aforementioned task mapping chromosome and using the MC security model, the chromosome (12,12,12,12,12,12,15,12,15) encodes a cryptographic configuration in which all intra-device communication uses 128-bit/10-round Rijndael encryption (security level 12) level and all inter-device communication uses 409-bit ECC (security level 15).

For DVFS, the *F* chromosome specifies the frequency (in MHz) at which each task executes. The voltage itself is not defined within the chromosome, as it can be inferred from the task assignment and frequency. The task ordering for the *F* chromosome is the same as the *M* chromosome. Consider the *F* chromosome of (300,300,700,700,300,300,100,1200,1200,100) and M chromosome of (0,0,0,0,0,0,2,1,1,2).

The *F* chromosome indicates the R2G task has a frequency of 300 MHz, and the *M* chromosome indicates the R2G task is mapped to software execution on ED1. The required voltage setting for executing R2G at 300 MHz on ED1's processor can then be determined.

The initial population generation randomly assigns task mappings, cryptographic implementations, and frequency settings for each task within the dataflow model for each member of the initial population. However, the initial population generation ensures all configurations meet the specified constraints. For example, if a task is mapped to hardware on ED6, the randomly-generated frequency for that task will be between 10 MHz and 150 MHz.

During selection and crossover, parents are selected proportional to their fitness, and a crossover probability of 0.6 is used (i.e., there is a 60% chance the parent's chromosomes are crossed over to produce the two children and a 40% chance the parent's chromosomes are copied to the children). The crossover point is randomly selected using a uniform distribution. Each mutation randomly changes one task mapping, cryptographic assignment, or frequency assignment with a 5% probability.

Genetic optimization algorithms typically begin by randomly generating an initial population in which some population members may violate system constraints [34]. To ensure that the final optimized implementation yields primarily population members that do not violate constraints, the system fitness evaluation function must incorporate penalties such as those for constraint violations. Alternatively, a genetic optimization algorithm could ensure all population members meet all the system constraints, but such an approach would require significant effort to initially explore a large portion of the design space to find enough such configurations [25], which is counterproductive to efficient optimization. Additionally, if penalty functions are not considered, individuals that violate one of the constraints are immediately rejected, and no information can be ascertained from those infeasible individuals, which can limit the design space that is evaluated. An alternative approach is to incorporate penalty functions [35,36] that penalize the system fitness for configurations that do not meet the constraints.

We utilize a hybrid approach, in which some constraints are mandated for all population members, and other utilize penalty functions in evaluating the overall system fitness. Specifically, constraints on hardware resources and end-to-end latency are strictly enforced, but constraints on energy consumption and minimum security level are not. Thus, we define two penalty functions: an energy penalty (EP) function and a minimum security level penalty (MSLP) function.

Algorithm 1 presents the pseudocode for our system fitness evaluation function with penalty constraints, where P is the population, MSL_C is the minimum security level constraint, E_C is the energy constraint, and SL_{MAX} is the maximum possible security level. SL_{MAX} is 11 for the MC security model, and 21 for the MCR security model. Since our aim is to find a feasible optimum solution, we can choose to penalize infeasible individuals. In other words, we extend the domain of the base system fitness function $f_{\mathrm{BASE}}(x)$ to determine an adjusted total fitness function $f(x)$. For the base fitness, we calculate the average security level $AvgSL(x)$ for each member of the population (line 3). However, the base fitness function only considers the optimization criteria, and not the system constraints. Hence, we calculate the total fitness $f(x)$ with the penalty functions (line 16).

To calculate the penalty for the energy constraint, the algorithm determines if each population member's energy consumption $E(x)$ is greater than the energy constraint E_C. If the energy consumption is greater than the energy constraint, a weighting function $f_E(x)$ calculates the ratio of the energy consumption to the difference of the maximum energy consumption of all population member and the energy constraint (line 5). For the minimum security level constraint, the weighting function $f_{MSL}(x)$ calculates how far the population's member minimum security level $MinSL(x)$ is to the minimum security level constraint MSL_C. Quadratic functions are used to calculate the energy penalty $\mathscr{P}_{\mathrm{E}}(x)$ (line 6) and the minimum security level penalty $\mathscr{P}_{\mathrm{MSL}}(x)$ (line 12) for the overall system fitness (line 6). Finally, the total fitness function $f(x)$ is the sum of the base system fitness and penalties (line 16).

Algorithm 1: Fitness function with penalty

Input: P, MSL_C, E_C.
Output: $f(x)$.

E_{MAX} = argmax of E(x)
  for $x \in P$ **do**
    $f_{\text{BASE}}(x) = AvgSL(x)/SL_{MAX}$
    if ($E(x) > E_C$) **then**
      $f_E(x) = E(x)/E_{MAX} - E_C$
      $\mathscr{P}_{\text{E}}(x)$ = sqrt ($6 \times f_E(x)$)/6
    else
      $\mathscr{P}_{\text{E}}(x) = 0$
    end if
    if ($MinSL(x) < MSL_C$) **then**
      $f_{MSL}(x) = MSL_C - MinSL(x)/MSL_C$
      $\mathscr{P}_{\text{MSL}}(x)$ = sqrt ($6 \times f_{MSL}(x)$)/6
    else
      $\mathscr{P}_{\text{MSL}}(x) = 0$
    end if
      $f(x) = f_{\text{BASE}}(x) - (\mathscr{P}_{\text{E}}(x) + \mathscr{P}_{\text{MSL}}(x))$
    end for

4.6. Security Policy Constraints

The use of a mixed cryptography model enables the specification of constraints for implementing specific security policies [37]. For example, a design may define a security policy in which all inter-device communication must use asymmetric cryptography algorithms. Such security policy constraints are supported within the security-driven optimization process as hard constraints within the genetic optimization. To evaluate the integration of security policy constraints within the optimization process, we consider three different specific constraints.

Security policy constraint 1 (C1) requires asymmetric cryptography for all inter-device communication and requires symmetric cryptography for all intra-device communication. Using the MC and MCR security models, inter-device communication must use either ECC or RSA, while intra-device communication must use Rijndael encryption, 2 TDEA or 3 TDEA. Such a security policy is useful to ensure that all inter-device communication, which is less secure then intra-device communication, uses asymmetric cryptography.

Security policy constraint C2 requires that each unique communication type (i.e., HW-SW, SW-SW, SW-HW, and HW-HW) uses a consistent cryptographic algorithm, but may use varying key sizes. For example, using the MC security model, HW-SW communication may select ECC at the cryptographic algorithm. In doing so, each unique channel using HW-SW communication can select security levels 3, 6, 10, or 15, which are different key size restrictions for ECC. This security policy is useful for reducing the complexity of key management.

Finally, security policy constraint C3 requires that each unique communication type uses a consistent cryptographic algorithm and key size.

5. Experimental Results

5.1. Experimental Setup

To evaluate the proposed security-driven optimization methodology, in addition to the video-based object detection and tracking application, we consider three additional applications with differing computational and communication requirements, which are also representative of dynamic

workloads. The video-based vehicle detection and tracking application, shown in Figure 2, represents an application with high computation and high communication (HCHC). Additionally, using TGFF [38], we generated synthetic applications representing a high-computation, low-communication (HCLC) application, a low-computation, high-communication (LCHC) application, and a low-computation, low-communication (LCLC) application. Figure 4 shows the resulting application models for the three generated applications. Table 3 summarizes the characteristics of the four applications, specifying the number of tasks, connectivity, average latency, average communication requirements, speedup for hardware-based implementation of software tasks, and power consumption for the hardware implementations. We further defined an energy constraint for each application, reported in Table 3.

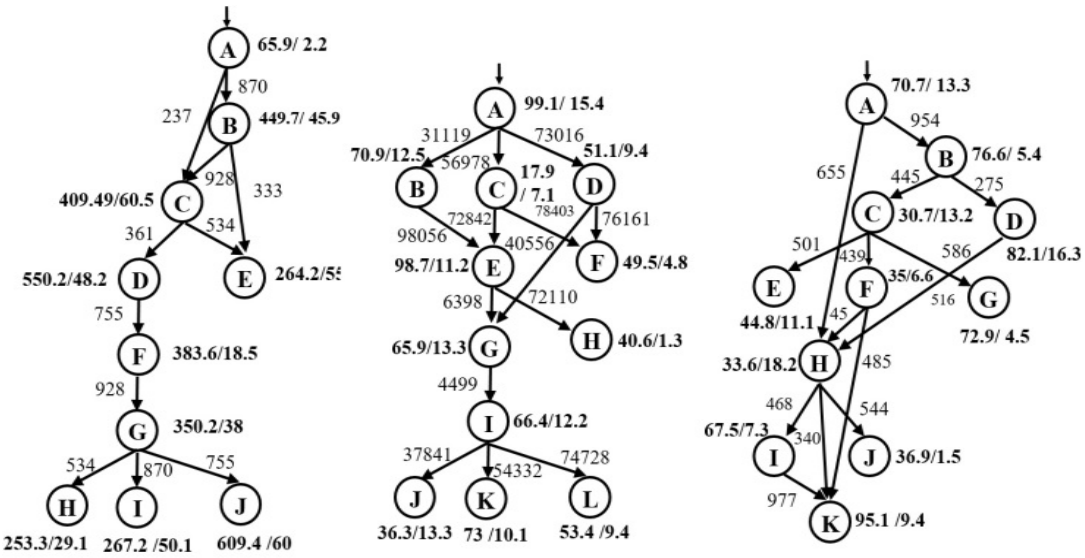


Figure 4. Task graphs for three synthetic applications (HCLC, LCHC, and LCLC).

Table 3. Characteristics of VBODT and synthetic applications.

	VBODT (HCHC)	HCLC	LCHC	LCLC
Tasks	10	10	12	11
Connectivity (edges/node)	0.9	1.1	1.16	1.27
Avg. Latency (ms)	357 (0.04, 2.8 K)	360 (66, 609)	60.2 (18, 99)	58 (31, 95)
Avg. Comm. (words)	177 K (10, 307 K)	645 (237, 928)	55 K (4.5 K, 98 K)	509 (45, 977)
HW Speedup	13 X (0.6–12 X)	10 X (2–60 X)	6 X (1–15 X)	6 X (1–15 X)
HW Static Power (mW)	78.1 (14.8, 123)	78.1 (14.8, 123)	78.1 (14.8, 123)	78.1 (14.8, 123)
HW Dynamic Power (mW)	125.8 (14.9, 785)	125.8 (14.9, 785)	62.9 (7.5, 392.5)	62.9 (7.5, 392.5)
Energy Constraint	10 J	0.5 J	1 J	0.5 J

The size of the design space is a combinatorial function of the number of tasks in the application, communication channels, unique hardware and software implementation supported by the target distributed embedded architecture, unique cryptographic implementations supported in the security model, and the available DVFS settings for the software and hardware implementations. For N tasks, M communication channels, C cryptographic implementations, A hardware and software implementation options, and F DVFS settings, the total design space is. $A^{N\cdot}\ C^{M\cdot}\ F^{N}$. For example,

for the video-based object detection and tracking application, architecture A1, and MC security model, the total design space size is $3^{10} \cdot 21^{9} \cdot 12^{10} = 2.9 \times 10^{27}$.

5.2. Genetic Algorithm Performance

Figure 5 presents the performance of the genetic optimization algorithm for increasing number of generations for the HCHC application targeting the A1 architecture using an energy constraint of 50 J and a minimum-security level constraint of 0. The vertical axis shows the best average security level across generations of the genetic optimization, starting from the initial population generation to 100 generations. The greatest increases in the average security level are achieved within the first 30 generations, in which the average security level increases by 12.6%. Across the next 50 generations, the average security level increases another 3.0%. For this application and target architecture, the security level does not increase beyond 80 generations, although the genetic optimization algorithm uses 100 generations.

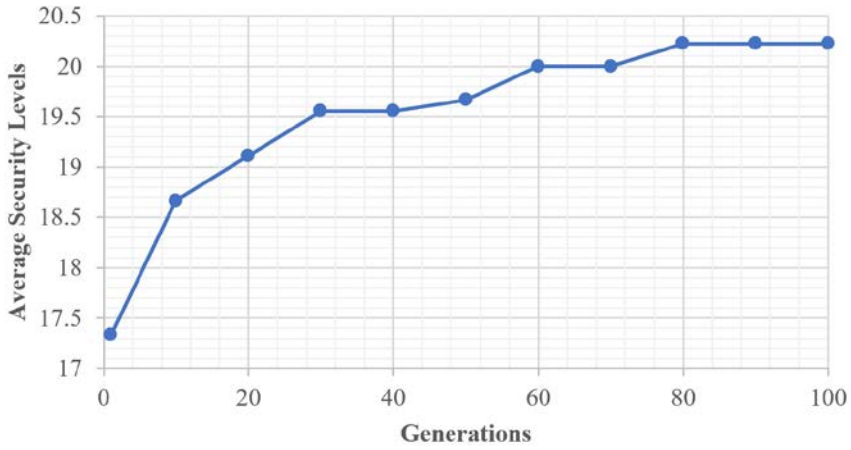


Figure 5. Performance of the genetic algorithm for the HCHC application, A1 architecture with DVFS, an energy constraint of 50 J, and a minimum security level constraint of 0.

5.3. MC/MCR Security Model

In evaluating the four applications and four distributed embedded architectures, we compared the proposed mixed cryptography model to that of using a single cryptographic (SC) implementation, specifically comparing to a model using only Rijndael encryption with different key sizes and rounds. Furthermore, we conducted experiments both with and without DVFS, to demonstrate the increased security afforded by using DVFS when subject to an energy constraint. Figure 6 presents the (a) average security strength (in equivalent Rijndael encryption key size) and (b) end-to-end latency (in seconds) for the HCHC application, using the SC and MC models, and targeting all four distributed embedded architectures, but without using DVFS. Energy constraints range from 30 to 130 J. For some architectures and energy constraints, no feasible implementation exists. For example, the average security strength for SC (A3) is infeasible until the energy constraint is 70 J. Across all architectures, the mixed cryptography approach achieves higher security strength. For architecture A1, MC increases the security strength by 49.9 bits, on average. Overall, the MC model using architecture A1 achieves the highest average security strength of 278.2 bits. Notably, there are several inflection points at which different architectures achieve a higher average security strength or lower latency, which demonstrate the tradeoffs between different architectural configurations. For some architectures and energy constraints, no feasible configurations exist. For example, for architecture A2, no feasible configuration is possible when the energy constraint is less than 50 J. However, for architecture A1 and A3, using MC yields feasible configurations with lower energy constraints than when using SC.

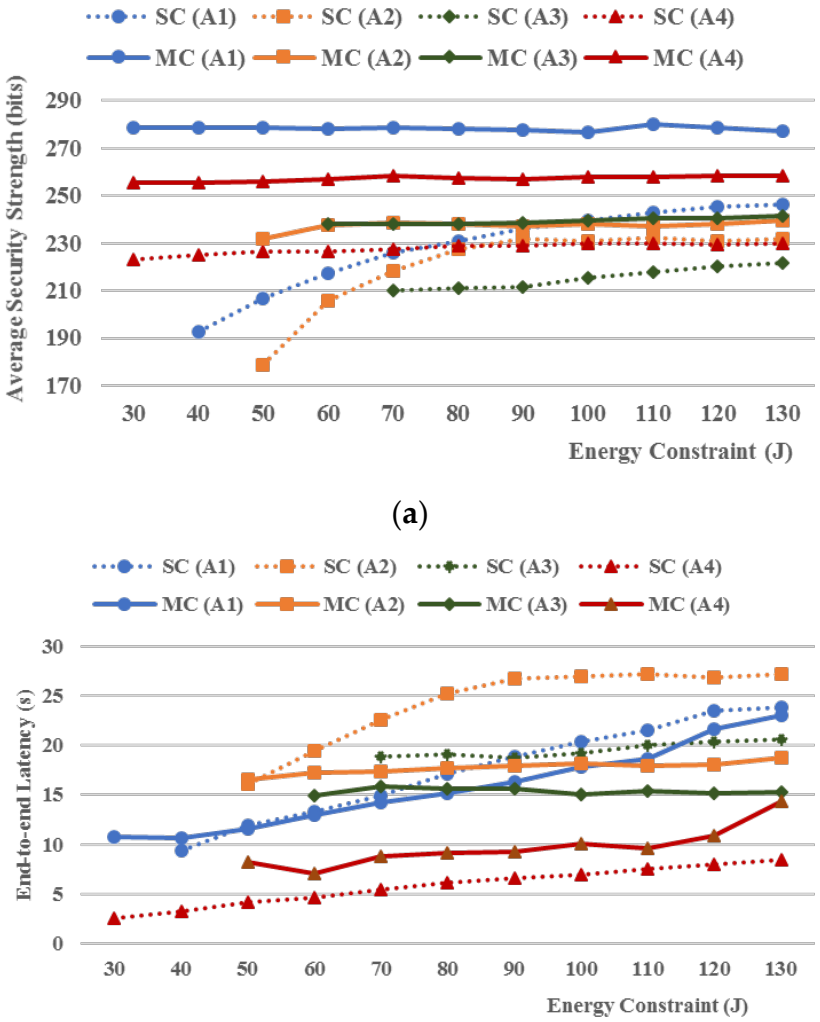


Figure 6. Average security strength and end-to-end latency for SC and MC for the HCHC application for all distributed embedded architectures.

The effect of using MC on the end-to-end-latency depends on the architecture and energy constraints. For architectures A1, A2, and A3, MC results in lower latency for all but one energy constraint. However, for A4, MC results in increased latency, with an increase of 67.7% on average. For this architecture, in order to achieve the increased security, the optimization utilizes ECC-571, which has the highest security level, but incurs a significant increase in latency.

To compare the benefits of the mixed cryptography model and DVFS, we compare the MC, MCR, and SC models for the HCHC application targeting architecture A1. Figure 7 presents the normalized average security strength achieved using MC, MCR, MC with DVFS, and MCR with DVFS, compared to the average security level of the SC model. MCR with DVFS achieves the highest average security strength, with an increase of 55.32 bits (or 23.2%) compared to the SC. At the other extreme, MC without DVFS increases the average security strength by 40.7 bits (or 17.8%) compared to the SC. For the mixed cryptography implementations, the restricted MCR model increases the security strength by up to 10% compared to the MC.

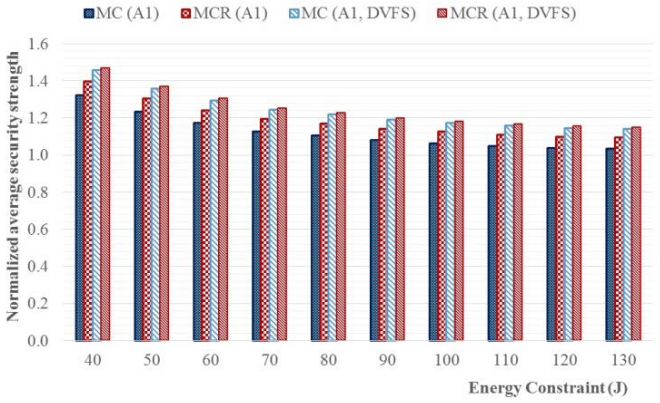
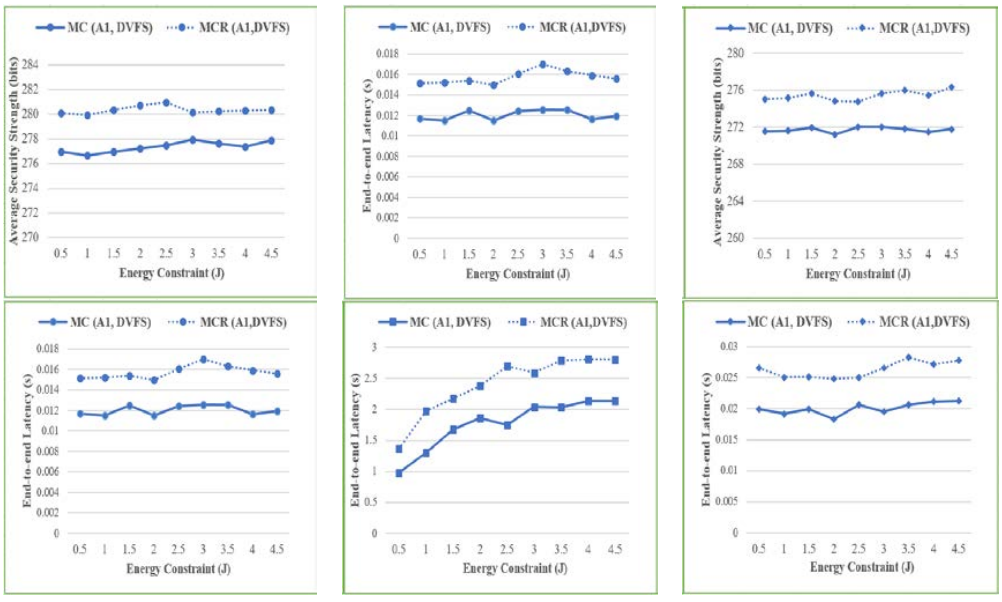


Figure 7. Normalized average security strength for MC, MCR, MC with DVFS, and MCR with DVFS, normalized to average security level of the SC security model.

Figure 8 presents the average security strength and end-to-end latency for the HCLC, LCHC, and LCLC applications using architecture A1. Across all applications, and all but one energy constraint (i.e., E_C = 0.5 J for LCHC), MCR yields a higher security, with increases up to 5.09%. For tasks with low communication requirements (HCLC and LCLC), the increase in security is fairly consistent across different energy constraints, which is a result of the relatively small impact communication has on the end-to-end latency. Since the computational latency for the applications is the dominant component, the use of DVFS can be applied even when energy constraints are tight. Importantly, the communication latency is determined by either the wireless communication protocol (e.g., inter-device communication) or the system bus (e.g., intra-device communication), neither of which are affected by the use of DVFS. Thus, for applications with high communication, the use of MCR requires a longer latency, but the use of DVFS has a limited effect in reducing the energy consumption. Thus, for tight energy constraints, MCR does not yield as much improvement in the security strength.



(**a**) HCLC (**b**) LCHC (**c**) LCLC

Figure 8. Average security strength and end-to-end latency for (**a**) HCLC; (**b**) LCHC; and (**c**) LCLC application for architecture A1.

5.4. Mixed Cryptography Security Model with Security Policy Constraints

To evaluate the impact of security policies, we analyzed the increased security strength for the MC and MCR security models with security policy constraint C1 compared the base SC model without a constraint. Figure 9 presents the increase in average security strength of MC and MCR for the HCHC, HCLC, LCHC, and LCLC applications targeting architectures A1, A2, A3, and A4. Notably, both MC and MCR with the security policy constraint C1 yields increased security across all applications and all architectures. On average, MC yields an increase of 13.2%, and MCR yields an increase of 12.7%. In the best case, MC and MCR yield improvements of 120% and 55%, respectively. In some cases, MCR yields higher security than MC (e.g., HCHC on architecture A4) with increases up to 44.3%. However, in more cases, MC yields higher security (e.g., LCLC on A2, A3, and A4, and HCLC and LCLC on A3). We highlight several key observations below.

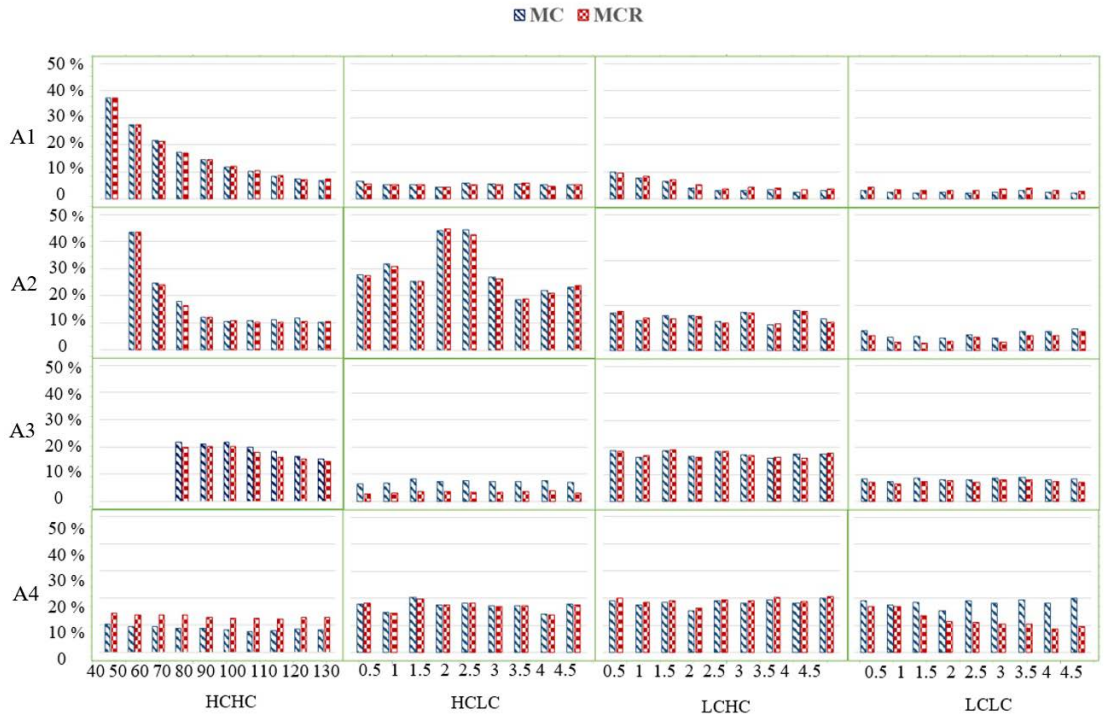


Figure 9. Percentage improvement of average security strength of MC and MCR, compared to SC, with security policy constraint C1 for HCHC, HCLC, LCHC, and LCLC applications (rows) and the four distributed heterogeneous architectures A1, A2, A3, and A4 (columns).

For low-communication tasks (i.e., HCLC and LCLC), MC typically yields a higher average security strength. Low communication applications tend to have less inter-device communication. As such, the increased latency required by the asymmetric cryptography for inter-device communication is less than in other applications. This in turn allows DVFS to be used more aggressively to reduce energy consumption. For LCLC on architecture A4, this increase in security is 52.1%.

For the HCHC application on architectures A1 and A2, the benefits of MC and MCR cryptography decrease and the energy constraint increases. This decrease is not a direct effect of the MC or MCR model or the security policy constraint. Instead, it is primarily due to the high level of security the SC model can achieve when the energy constraint is relaxed. Specifically, the average security strength for MC or MCR with security policy constraint C1 is consistent across different energy constraints, whereas the average security for SC increases linearly with the increased energy constraint. Thus, relative to SC, the advantage of MC and MCR is not as significant.

For HCLC on architecture A2, the increase in average security strength varies across the different energy constraints, with increases ranging from 18.6% to 44.3%. Notably, the greatest increase in security is achieved for energy constraints of 2–2.5 J, which are in the middle of the constraints considered. These results can primarily be attributed to the lack of an ED device supporting parallel execution of tasks, which results in the optimization finding very different configurations that meet the energy and security constraints. In other words, the number of feasible configuration for this architecture and application is small, so the genetic algorithm requires more effort to find feasible configurations.

Compared to MC without a security policy constraint, using security constraint C1 results in a 159% increase in latency, on average. This increase is mainly attributed to the requirement to asymmetric cryptographic implementation for inter-device communications, which increase the latency for the communication methods with the longest latency. Without the security policy constraint, asymmetric cryptography is utilized for only 24.5% of the inter-device communication for MC, compared to 100% with the constraint. However, using MCR with the security policy constraint results in a 27% decrease in end-to-end latency. This is due to two reasons. First, MCR without the constraint already uses asymmetric cryptography for 48.5% of inter-device communication, on average, so the impact of the security policy is diminished compared to MC. Second, symmetric cryptography can reduce the latency for intra-device communication, yielding an overall decrease will still achieve a higher average security strength.

In a few cases, MC and MCR with security policy constraint C1 decrease the end-to-end latency, even compared to the base SC case. To illustrate this case, Figure 10 presents the increase in end-to-end latency of MC and MCR with security constraint C1 compared to SC for application LCLC on architecture A3. Using MC, asymmetric cryptography is used for only 27.3% of the communication channels. Combined with the fact that inter-device communication is lower than other applications, the impact of using asymmetric cryptography is minimal, and the optimization algorithm is able to optimize both average security and latency relative to SC. In contrast, for MCR, as the energy constraint is relaxed, the end-to-end latency increases. From our observations, for tight energy constraints, inter-device communication is rare, so, again, asymmetric cryptography is used infrequently (e.g., 9% of the communication channels). As the energy constraint is relaxed, asymmetric algorithms are used more often, as much as 27.3%, which results in a significant increase in end-to-end latency.

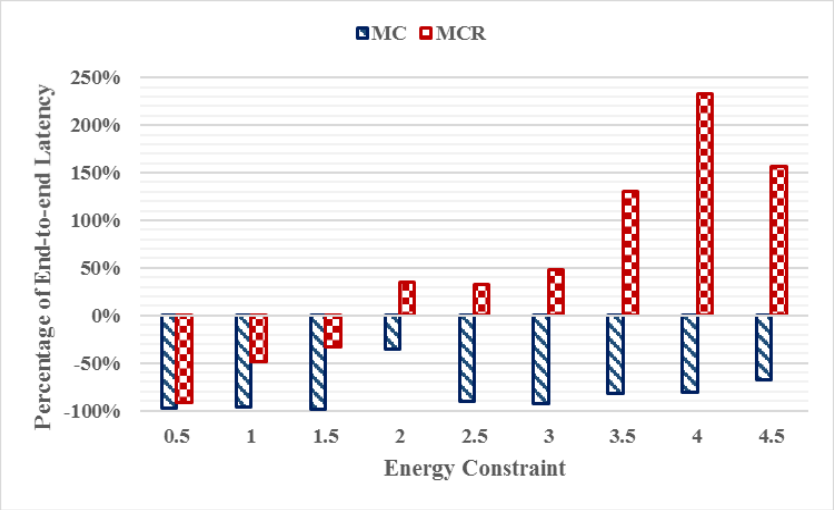


Figure 10. Percentage change in end-to-end latency of MC and MCR with security policy constraint C1 compared to SC for the LCLC application on architecture A3.

Finally, we sought to understand the impact of the different security models and the three security policy constraints on the resulting diversity of the cryptographic implementations used. Figure 11 presents how many cryptographic algorithms and how many different key sizes are used based on SC, MC, and MCR security models with security policy constraints C1, C2, and C3, averaged across all applications and architectures. The diversity of the SC security model is very low, on average using only 1.5 different key sizes. Both MC and MCR without security policy constraints result in using slightly more than two different cryptographic implementations, with MCR yielding a significant increase in the distinct number of key sizes used.

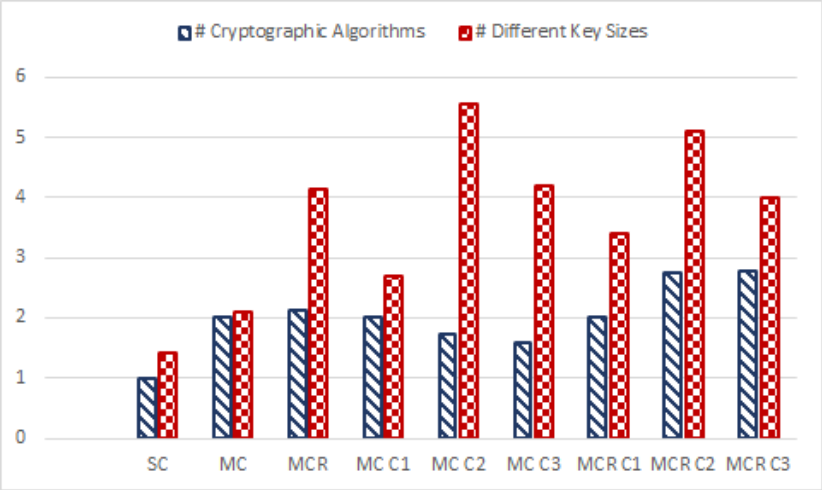


Figure 11. Diversity of cryptographic implementations, reported as the number of distinct cryptographic algorithms and key sizes used, for SC, MC, and MCR security models with security policy constraints C1, C2, and C3.

The security policy constraints can have different impacts on the cryptographic diversity depending on the security model used. In some cases, the policy may force the optimization to use a greater number of cryptographic algorithms and, in other cases, it can reduce the diversity. Security policy constraint C1 not only yields an increase in diversity of the cryptographic algorithms but also increases the diversity of key sizes, with increases of 88.6% and 138.7% for MC and MCR, respectively.

Using security policy constraint C2 with MC yields lower diversity in the cryptographic algorithm but the highest diversity if key sizes. However, for MCR, security policy constraint C2 yields increased diversity in both algorithms and key size. MCR with C2 uses 2.75 different cryptographic algorithms and 5.1 different key sizes. Only MCR with security policy constraint C3 yields higher diversity in the cryptographic algorithms (2.13 vs. 2.77), but with lower diversity in key sizes (4.13 vs. 4).

6. Conclusions and Future Work

This paper presented a modeling and optimization framework for adaptive, distributed, reconfigurable, and heterogeneous embedded systems. Our modeling framework supports the efficient and robust modeling of applications, architectures, mixed cryptographic implementations, and security policy constraints. To support the analysis and evaluation of mixed cryptographic implementations, we present a level-based security metric for specifying a relative ranking of the available implementations. Our experimental results demonstrate that mixed cryptographic implementations yield increased security compared to using a single cryptographic algorithm, with increases in the average equivalent key size (up to 45%). Using several representative benchmarks, we further analyzed and highlighted the applications types and architectures for which the MC and MCR security models are most suited. For HCHC application with architecture A1, MCR yields higher security (up to 45.4% improvement), whereas for HCLC application with architecture A2,

MC yields higher security (up to 38.03% improvement). Lastly, imposing security policy constraints can yield increased security strength (up to 44.7%) and increased diversity (up to 289%) for the same energy constraint.

Future work includes analyzing the impact diversity in cryptographic implementation afforded by the mixed cryptographic approach, and its overall effect on system security. As the use of mixed cryptographic implementations will require more complex key management schemes, we further seek to integrate key management within the system level optimization framework. Future work also includes adapting the proposed self-aware optimization framework to integrate dynamic profiling methods and runtime adaptive security policies. As the heterogeneous components integrated within the embedded devices within the distributed architecture will impact the overall performance, energy consumption, and security, future work includes investigating the use of automated exploration [9,10] of the heterogeneous resources within the embedded devices.

Author Contributions: H. Nam and R. Lysecky conceived and designed the experiments; H. Nam performed the experiments; H. Nam and R. Lysecky analyzed the data and wrote the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, X.; Wang, S.; Bi, D. Distributed Visual-Target-Surveillance System in Wireless Sensor Networks. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2009**, *39*, 1134–1146. [CrossRef] [PubMed]
2. Jahnavia, V.; Ahameda, S. Smart Wireless Sensor Network for Automated Greenhouse. *IETE J. Res.* **2015**, *61*, 180–185. [CrossRef]
3. Rifà-Pous, H.; Herrera-Joancomartí, J. Computational and Energy Costs of Cryptographic Algorithms on Handheld Devices. *Future Internet* **2011**, *3*, 31–48. [CrossRef]
4. Gu, Z.; Han, G.; Zeng, H.; Zhao, Q. Security-Aware Mapping and Scheduling with Hardware Co-Processors for FlexRay-based Distributed Embedded Systems. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 3044–3057. [CrossRef]
5. Zhang, X.; Zhan, J.; Jiang, W.; Ma, Y.; Jiang, K. Design Optimization of Energy- and Security-Critical Distributed Real-Time Embedded Systems. In Proceedings of the IEEE International Symposium on Parallel & Distributed Processing Workshops and PhD Forum, Cambridge, MA, USA, 20–24 May 2013; pp. 741–750.
6. Medien, Z.; Machhout, M.; Bouallegue, B.; Khriji, L.; Baganne, A.; Tourki, R. Design and Hardware Implementation of QoSS-AES Processor for Multimedia applications. *Trans. Data Priv.* **2010**, *3*, 43–64.
7. Pop, P.; Eles, P.; Peng, Z.; Pop, T. Analysis and Optimization of Distributed Real-Time Embedded Systems. *ACM Trans. Des. Autom. Electron. Syst.* **2006**, *11*, 593–625. [CrossRef]
8. Selicean, D.; Pop, P. Design Optimization of Mixed-Criticality Real-Time Embedded Systems. *ACM Trans. Embed. Comput. Syst.* **2015**, *14*, 50.
9. Pomante, L. System-level Design Space Exploration for Dedicated Heterogeneous Multi-Processor Systems. In Proceedings of the IEEE International Conference on Application-specific Systems, Architectures and Processors, Sousse, Tunisia, 22–24 June 2011; pp. 79–86.
10. Pomante, L. HW/SW Co-Design of Dedicated Heterogeneous Parallel Systems: An Extended Design Space Exploration Approach. *IET Comput. Dig. Tech.* **2013**, *7*, 246–254. [CrossRef]
11. Shang, L.; Dick, R.P.; Jha, N.K. SLOPES: Hardware–Software Co-synthesis of Low-Power Real-Time Distributed Embedded Systems with Dynamically Reconfigurable FPGAs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2007**, *26*, 508–526. [CrossRef]
12. Lin, C.-W.; Zhu, Q.; Sangiovanni-Vincentelli, A. Security-Aware Mapping for TDMA-Based Real-Time Distributed Systems. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2–6 November 2014; pp. 24–31.
13. Lin, C.-W.; Zhu, Q.; Sangiovanni-Vincentelli, A. Security-Aware Modeling and Efficient Mapping for CAN-Based Real-Time Distributed Automotive Systems. *IEEE Embed. Syst. Lett.* **2015**, *7*, 11–14. [CrossRef]
14. Jiang, K.; Eles, P.; Peng, Z. Co-Design Techniques for Distributed Real-Time Embedded Systems with Communication Security Constraints. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 12–16 March 2012; pp. 947–952.

15. Singh, G.; Kinger, S. Integrating AES, DES, and 3-DES Encryption Algorithms for Enhanced Data Security. *Int. J. Sci. Eng. Res.* **2013**, *4*, 7.
16. Mansour, I.; Chalhoub, G. Evaluation of different cryptographic algorithms on wireless sensor network nodes. In Proceedings of the 2012 International Conference on Wireless Communications in Unusual and Confined Areas (ICWCUCA), Clermont Ferrand, France, 28–30 August 2012.
17. Peter, S.; Zessack, M.; Vater, F.; Panic, G.; Frankenfeldt, H.; Methfessel, M. Encryption-Enabled Network Protocol Accelerator. In Proceedings of the International Conference on Wired/Wireless Internet Communications, Tampere, Finland, 28–30 May 2008; pp. 79–91.
18. Kuppuswamy, P.; Al-Khalidi, S. Hybrid Encryption/Decryption Technique Using New Public Key and Symmetric Key Algorithm. *Int. J. Inf. Comput. Secur.* **2014**, *6*, 372–382. [CrossRef]
19. Xin, M. A Mixed Encryption Algorithm Used in Internet of Things Security Transmission System. In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Xi'an, China, 17–19 September 2015.
20. Ravi, S.; Raghunathan, A.; Kocher, P.; Hattangady, S. Security in Embedded Systems: Design Challenges. *ACM Trans. Des. Autom. Electron. Syst.* **2004**, *3*, 461–491. [CrossRef]
21. Verbelen, Y.; Braeken, A.; Kubera, S.; Touhafi, A.; Vliegeny, J.; Mentens, N. Implementation of a Server Architecture for Secure Reconfiguration of Embedded Systems. *ARPN J. Syst. Softw.* **2011**, *1*, 270–279.
22. Hwang, D.; Schaumont, P.; Tiri, K.; Verbauwhede, I. Securing Embedded Systems. *IEEE Secur. Priv.* **2006**, *4*, 40–49. [CrossRef]
23. Xiao, K.; Forte, D.; Jin, Y.; Karri, R.; Bhunia, S.; Tehranipoor, M. Hardware Trojans: Lessons Learned after One Decade of Research. *ACM Trans. Des. Autom. Electron. Syst.* **2016**, *22*, 6. [CrossRef]
24. Fern, N.; San, I.; Koç, Ç.K.; Cheng, K.-T. Hiding Hardware Trojan Communication Channels in Partially Specified SoC Bus Functionality. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2017**, *36*, 1435–1444. [CrossRef]
25. Nam, H.; Lysecky, R. Latency, Power, and Security Optimization in Distributed Reconfigurable Embedded Systems. In Proceedings of the Reconfigurable Architecture Workshop (RAW), Chicago, IL, USA, 23–27 May 2016; pp. 124–131.
26. Sandoval, N.; Mackin, C.; Whitsitt, S.; Lysecky, R.; Sprinkle, J. Runtime Hardware/Software Task Transition Scheduling for Runtime-Adaptable Embedded Systems. In Proceedings of the International Conference on Field-Programmable Technology (ICFPT), Kyoto, Japan, 9–11 December 2013; pp. 342–345.
27. Najib, A.K. An Empirical Study to Compare the Performance of some Symmetric and Asymmetric Ciphers. *Int. J. Secur. Appl.* **2013**, *7*, 1–16.
28. Kim, H.; Lee, S. Design and Implementation of a Private and Public Key Crypto Processor and Its Application to a Security System. *IEEE Trans. Consum. Electron.* **2004**, *50*, 214–224.
29. Iana, V.; Anghelescu, P.; Serban, G. RSA encryption algorithm implemented on FPGA. In Proceedings of the International Conference on Applied Electronics (AE), Pilsen, Czech Republic, 7–8 September 2011.
30. NIST Special Publication 800-57. *Recommendation for Key Management, Part 1: General*; NIST Special Publication: Gaithersburg, MD, USA, 2016.
31. Li, H.; Huang, J.; Sweany, P.; Huang, D. FPGA implementations of elliptic curve cryptography and Tate pairing over a binary field. *J. Syst. Arch.* **2008**, *54*, 1077–1088. [CrossRef]
32. Jaervinen, K. The State-of-the-Art of Hardware Implementations of Elliptic Curve Cryptography. In Proceedings of the Workshop on Hardware Benchmarking, Bochum, Germany, 7 June 2017.
33. Alrimeih, H.; Rakhmatov, D. Fast and Flexible Hardware Support for ECC Over Multiple Standard Prime Fields. *IEEE Trans. Very Large Scale Integr. Syst.* **2014**, *22*, 2661–2674. [CrossRef]
34. Homaifar, A.; Qi, C.X.; Lai, S.H. Constrained optimization via genetic algorithms. *Simulation* **1994**, *62*, 242–254. [CrossRef]
35. Hilton, A.; Culver, T. Constraint-handling methods for optimal groundwater remediation design by genetic algorithms. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, USA, 14 October 1998; pp. 3937–3942.
36. Tessema, B.; Yen, G.G. An Adaptive Penalty Formulation for Constrained Evolutionary Optimization. *IEEE Trans. Syst. Man Cybern. A Syst. Hum.* **2009**, *39*, 565–578. [CrossRef]

37. Liu, X.; Zhao, M.; Li, S.; Zhang, F.; Trappe, W. A Security Framework for the Internet of Things in the Future Internet Architecture. *Future Internet* **2017**, *9*, 27. [CrossRef]
38. Dick, R.P.; Rhodes, D.L.; Wolf, W. TGFF: Task Graphs for Free. In Proceedings of the International Workshop on Hardware/Software Codesign, Seattle, WA, USA, 18 March 1998.