Energy Efficient Security Algorithm for Power Grid Wide Area Monitoring System

Meikang Qiu, *Senior Member, IEEE*, Wenzhong Gao, *Senior Member, IEEE*, Min Chen, *Senior Member, IEEE*, Jian-Wei Niu, *Member, IEEE*, and Lei Zhang, *Member, IEEE*

***Abstract*—Modern power grid is the most complex human-made system, which is monitored by *wide-area monitoring system* (WAMS). Providing time-synchronized data of power system operating states, WAMS will play a crucial role in next generation smart grid protection and control. WAMS helps secure efficient energy transmission as well as reliable and optimal grid management. As the key enabler of a smart grid, numerous sensors such as PMU and current sensors transmit real-time dynamic data, which is usually protected by encryption algorithm from malicious attacks, over *wide-area-network* (WAN) to power system control centers so that monitoring and control of the whole system is possible. Security algorithms for power grid need to consider both performance and energy efficiency through code optimization techniques on encryption and decryption. In this paper, we take power nodes (sites) as platforms to experimentally study ways of energy consumptions in different security algorithms. First, we measure energy consumptions of various security algorithms on CrossBow and Ember sensor nodes. Second, we propose an array of novel code optimization methods to increase energy consumption efficiency of different security algorithms. Finally, based on careful analysis of measurement results, we propose a set of principles on using security algorithms in WAMS nodes, such as cryptography selections, parameter configuration, and the like. Such principles can be used widely in other computing systems with energy constraints.**

***Index Terms*—Cryptographic algorithm, power grid, scheduling, security, smart grid, wide-area monitoring system.**

I. Introduction

THE MODERN power grid is the most complex human-made system, which is currently managed by the Supervisory Control and Data Acquisition System (SCADA) and energy management systems (EMS). Typical SCADA and EMS have slow data update rate and cannot meet performance demand of a smart grid.

Thanks to the rapid development of synchronized global positioning system (GPS), synchronized-measurement technology has been developed since the 1970s for emerging wide-area monitoring system (WAMS) [1]. WAMS is essentially a sensor network deployed over a vast geographical area overlaying the power network infrastructure. The backbone of this network is high speed Internet. Attempting to monitor the entire power system dynamics, WAMS synchronizes all monitored data by time-stamping GPS coordinated universal time (UTC) so as to globally validate measurements regardless of measuring locations. The key component in WAMS is the phasor measurement unit (PMU). Providing time-synchronized data of power system operating states, WAMS will play crucial role in next generation smart grid protection and control. WAMS will help secure efficient energy transmission as well as reliable and optimal grid management.

Security is a critical issue in the design and operation of WAMS. As the key enabler of a smart grid, numerous sensors such as PMU and current sensors transmit real-time dynamic data over a wide-area network (WAN) to power system control centers so that monitoring and control of the whole system is possible. In order to protect the system from malicious attacks, data is encrypted first before transferring through the network. At the destination, decryption will be used to get the original data. Since WAMS is an energy constrained system, we need to consider the energy consumption of the computation. Hence, new security algorithm design with the consideration of code optimization is critical to the power grid real-time operation in order to maintain the stability of the power systems.

For the IC hardware of a sensor, such as a PMU, its main power consumption includes dynamic power consumption, leakage power consumption, and short circuit power consumption. The dynamic power consumption, which accounts for 60% to 80% of the total power consumption, has a close relationship with the activities of a system [2], [3]. Meanwhile, enhancing the system security certainly increases the system activities while leading to additional power consumption. In other words, system security and system activities are positively correlated. Therefore, we need to consider the trade-off between system security and energy consumption when we enhance the security level for energy-constrained systems.

From the security enhancement aspect, hardware implementations are highly efficient and have high security strengths. However, the power consumption of hardware implementations is higher than that of software implementations. The reason is because hardware implementations of security enhancement

Manuscript received October 14, 2010; revised March 06, 2011; accepted June 05, 2011. Date of publication August 15, 2011; date of current version November 23, 2011. This work was supported in part by the NSFC 61071061, the University of Kentucky Start Up Fund; NAP of Korea Research Council of Fundamental S&T, IT R&D program of KCA 10913-05004; the State Key Lab of Software Development Env. Grant BUAA SKLSDE-2010ZX-13, NSFC 60873241, ASF 20091951020. Paper no. TSG-00163-2010.

M. Qiu is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506 USA (e-mail: mqiu@engr.uky.edu).

W. Gao (corresponding author) is with the Department of Electrical and Computer Engineering, University of Denver, Denver, CO 80210 USA (e-mail: Wenzhong.Gao@du.edu).

M. Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea (e-mail: minchen@ieee.org).

J. Niu is with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China (e-mail: niujianwei@buaa.edu.cn).

L. Zhang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSG.2011.2160298

lead to higher IC complexity, which causes the increase of static power consumption and short circuit power consumption. From the energy consumption aspect, the higher level of design can provide higher potential solution space for energy reduction and lower cost [2], [4]. Recently, there is a new trend: the focus of energy optimization research is transferring to energy optimization on application level. Energy optimizations on the behavior level has certain limitations due to the poor flexibility of hardware implementations. Thus, for an energy-constrained system, we can satisfy the security requirements using software implementations rather than hardware implementations.

Basic security algorithms, e.g., cryptography algorithms, are the fundamental parts of any security system. A wide variety of security protocols and standards are developed based on basic security algorithms. In security algorithms, users can set the values of some parameters, such as key length and application mode, which facilitate the energy optimization on the application level. Therefore, we believe that the balance between security and energy consumption can be achieved by properly using the existing security algorithms. Meanwhile, due to the high flexibility of software implementations, we can reduce the energy consumption of executing security algorithms through code optimizations. Furthermore, when determining the configuration and the selection of security algorithms, one has to consider not only the power consumption of CPUs, but also the impact of the implementation on other components. Thus, the objective of energy optimizations is to have: 1) longer system lifetime and 2) lower system total energy consumption. In the rest of this paper, we study the security algorithms implemented in an energy-constrained system on our WAMS platform. The reasons of using the WAMS platform as our platform are as follows.

- WAMS is a typical energy-constrained system. The complexity of the hardware/software functions in WAMSs has a large impact on the lifetimes of sensor nodes in WAMSs.
- Compared with traditional energy-constrained systems, WAMSs face larger challenges. The security requirement of WAMSs covers every aspect of system security, such as confidentiality, integrity availability, which makes it more suitable to study the characteristics of different security algorithms.
- Executing security algorithms on sensor nodes of a WAMS impacts not only the energy consumption of CPUs, but also the energy comsuption of other components.

In this paper, we take power nodes (sites) as platforms to experimentally study ways of energy consumptions in different security algorithms. First, we design a micro-power measurement circuit to measure energy consumptions of various security algorithms on CrossBow and Ember sensor nodes. Second, we propose an array of novel code optimization methods to increase energy consumption efficiency of different security algorithms. Finally, based on careful analysis of measurement results, we propose a set of principles on using security algorithms in sensor network nodes, such as cryptography selections, parameter configuration, and the like. Such principles can be used widely in other computing systems with energy constraints.

The arrangement of this paper is as the following: we first introduce the related work of this topic in Section II, followed by the analysis of some basic issues related to WAMS security and energy consumption in Section III. In order to provide the insight of the impact of security algorithms, we measure the energy consumptions of several well-known security algorithms on two kinds of WAMS nodes in Section IV. Then, we present a code optimization method to reduce energy consumption of security algorithms in Section V. Finally, in Section VI, we provide a set of principles on applying security algorithms to energy-constrained systems, such as WAMSs.

II. Related Work

A large amount of works have been done on energy consumption of security algorithms and protocols in common network environment, such as the energy consumption of the SSL protocol in PC networks, the energy consumption of WEP in Wi-Fi networks, etc. However, there are few research on the energy consumption of security algorithms in WAMSs.

Recently, most of the energy consumption studies related to WAMSs are based on simulations. The energy consumption is usually measured based on the CPU computation time and the number of data packets. However, this is a coarse grain approximation. In simulation aspect, network simulations, such as NS2, TOSSIMM [5], and Atemu [6], can properly simulate the behaviors of the network protocols, but they are not able to simulate the single node well. Thus, the method mentioned above is not suitable for our study on the energy consumption of security algorithms in WAMSs.

There are some instruction-level energy evaluation models for WAMSs, such as AEON [7], and PowerTOSSIM [8]. These two models first measure the currents of the sensor nodes, followed by partitioning the measured currents to different code segments and different components of sensor nodes. Finally, the energy consumption of a certain code segment or a certain component is calculated. However, these models are not suitable for commercial WAMS sensors, due to the fact that most of the manufacturers only provide software in the form of "black box." Even obtaining the source code, inserting instruction for measuring is not convenient.

To obtain the energy characteristics of security algorithms, there are some studies based on physical measurement. Wander *et al.* [9] has measured the energy consumption of the RSA and ECC on MICA2DOT sensor nodes. However, this method cannot be implemented with the whole version of code into the WAMS nodes. Gupta *et al.* [10] has pointed out that the size of memory consumption of standard code is close to 4 KB for cryptography algorithms such as DES. This means that the memory will not be enough to directly implement standard algorithms on some WAMS nodes.

WAMS is an autonomic network consisting of a large number of sensor nodes deployed in the monitoring area. The sensor nodes are connected by an ad hoc network and communicate with the sink through multihop, as shown in Fig. 1. In a WAMS, the sensor nodes are the basic parts of the implementation of information sensing and communication. Compared to other wireless network, the WAMS is a specific application oriented network, which has characteristics of large size and dynamic topology.

A sensor node in WAMS is a system with multifunction, such as data collection, computation, and communication. Compared

TABLE I
MAJOR SECURITY THREATS IN A WAMS

Attack type	Attack method
Channel attack	Monitor, intercept, and tamper the data packets in public channels.
	Data replay attack, repeatedly sending the data which has been sent before.
	Message insert attack, destroy the reading of messages and the internal control data.
Denial of service attack	Node collaboration attack, malicious node prevents messages from broadcasting to certain areas of the WAMS network [16].
	Channel congestion. The attacker consumes the bandwidth of the channel, disables the sending and receiving of other nodes.
	Energy consumption. The attacker repeatedly sends the data request messages, consuming the energy of the target node [17].
Node attack	The attacker controls some nodes, analyzes and modifies them, or obtains confidential information in them [16].
Attack from malicious nodes	Sybil attack. Use the multiply IDs to deceive other nodes, so that the malicious nodes become the routing node more easily. Then attack the target node combining with other attack methods.
	Node copy attack. Steal the ID information from a legal node, then attack other nodes [18].
Protocol attack	Includes: routing protocol attack, selective forwarding attack, wormhole attack, Hello Flood attack, deceiving reply attack, and attack targets on data fusion [19], [20].

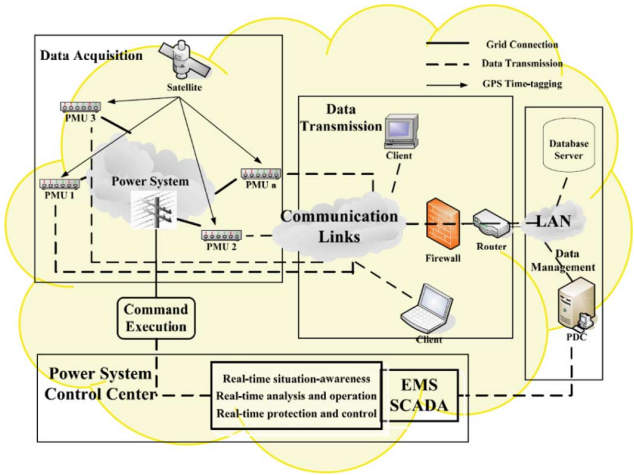


Fig. 1. The architecture of a WAMS.

to common sensor systems, the WAMS sensor nodes have their constraints as listed below.

- Poor computation power. The sensors usually adopt MCU that has slow computation speed as their processors.
- Limited memory space. The ATmega128L processor has only 4 KB SRAM onchip. And the high-end CrossBow Imote2 has 256 KB onchip and 32 MB offchip.
- Tight power consumption constraint. Due to the limitations of deployed area, cost, and physical size, some WAMS nodes are usually equipped with low capacity batteries. Thus, there are strict requirements for consecutive execution time on nodes.

Because of the strict constraints of computation resource and energy, the WAMS nodes can only execute simple computation tasks and communication tasks. Currently, the hardware/software codesign is a key part of WAMS study.

PMU-based WAMS is a cyber-physical system where the Internet-based communication network overlays the physical equipment-based power grid [11], [12]. Cybersecurity is crucial for ensuring integrity and resiliency of future smart grid. The encryption design for secure WAMS data communication must consider energy and bandwidth constraints.

PMU was first invented in Virginia Tech in 1988 to measure phasors of voltage and current, frequency and real/reactive power in real-time with GPS time tagging [1], [13]. PMUs have thus been continuously enhanced and are now being deployed in substations. The PMU data are collected in a phasor data concentrator (PDC) to facilitate real-time power system situation awareness, analysis, operation, protection, and control [13]–[15]. With large scale integration of variable renewable energy integration, PMU-based smart grid will ensure power system stability and reliability.

III. WAMS SECURITY ISSUES

A. Security Threats Encountered in WAMSs

WAMSs use public communication channels, in which every device inside or outside the network may obtain the information. The attackers can directly destroy the WAMS nodes due to the open deployment of nodes. Table I summarizes the major threats that WAMSs may encounter.

B. Energy Constraint on Security Algorithms

In order to prevent the attacker from deciphering directly on WAMS nodes, the data on nodes should be encrypted before stored. Complete verification information needs to be inserted into original message, so that the data packets will not be modified maliciously. Thus, it is necessary to introduce proper ID verification mechanism into WAMSs to defend ID-based attack such as Sybil and node duplication attack. Although the security threats encountered by WAMSs are diverse, data encryption, integrity protection, and verification are the most basic security service requirements in WAMSs. In WAMSs, the stronger the security algorithm is, the more energy consumption on the CPU [21]. Thus, in order to satisfy the security requirement on the WAMS, energy consumption is the major factor that constrains security algorithms.

C. Security Level and Energy Consumption

The limited energy and computation resources determine that the security mechanism in WAMS nodes should be implemented as simple as possible. Integrating security service directly in the WAMS nodes such as PMU is an efficient way to increase the security of nodes. The security requirement can be realized by three basic security services, i.e., encryption, integrity protection, and verification. Therefore, based on the security service integrated into the nodes, we can classify the

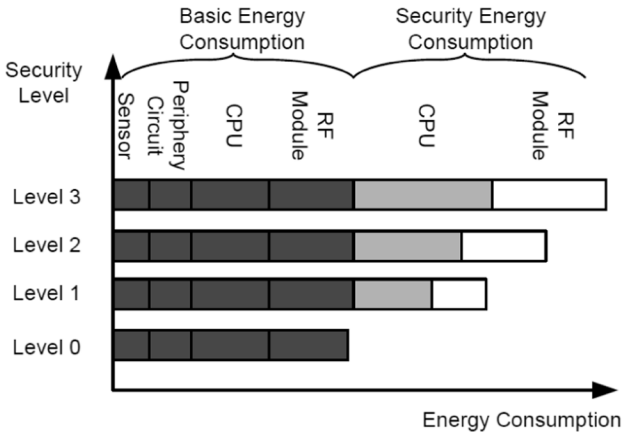


Fig. 2. The security levels and the distributions of energy consumption on a WAMS node.

security of a node into four levels: Level 0, nonsecurity service; Level 1, single security service; Level 2, double security services; and Level 3, triple security services. Note that even though multiple security schemes can achieve the same level of security, their energy consumptions are different.

The energy consumption of a node can be partitioned into two parts, when the security services are integrated into this node: 1) basic energy consumption, which relates to the application functions; 2) energy consumption caused by the security services. From the system stand point, the energy consumption of a WAMS node is mainly from the sensing circuit, the periphery circuit, the microprocessor, and the RF circuit.

The security algorithms are usually implemented by CPU on computation-intensive tasks, which cause the increase of the CPU's energy consumption. Besides, the integrity protection and the verification service increase message lengths, leading to the increase of energy consumption on the RF transmit module. The security service has less impact on energy consumptions on sensors and periphery circuits. The energy consumption distribution across the node and the relationship between energy consumption and security level are shown in Fig. 2. The security levels are a set of discrete values, and the energy consumption is linear to the value of security level. Strictly speaking, the energy consumption is related to the complexity of the security service implementation [22].

Due to the difference among the computational power, different kinds of WAMS nodes can support algorithms with different levels of security strength. The energy consumption is related to the computational power of the node. For the nodes with low computational power, integration of the security algorithm will increase the computation time and the communication time, causing significant increase of the total energy consumption. On the other hand, for the nodes with strong computational power, the energy consumption on the CPU will not vary significantly. Furthermore, increasing the data transmission speed will help reduce the energy consumption of the node.

IV. Energy Consumption Measurement of Security Algorithms

In this section, we conduct the power consumption measurement for some security algorithm and scheme on the CrossBow and Ember nodes. And we also analyze the factors that impact the energy consumption. As we mentioned before, the energy consumption of a node is mainly caused by the CPU, the RF module, the sensor, and the periphery circuit [23]. In this section, we will analyze the energy consumption on the CPU and the RF module, due to the direct impact of the execution of security algorithms on the energy consumption.

TABLE II
Energy Consumptions on CrossBow MICA2 Nodes

The length of messages (bytes)	E_CPU (μJ)	E_RF (μJ)	E_Sum (μJ)
8	22	952	974
16	23	1220	1243
24	26	1462	1488
32	28	2409	2437

TABLE III
Energy Consumptions on Ember Nodes

The length of messages (bytes)	E_CPU (μJ)	E_RF (μJ)	E_Sum (μJ)
8	22	82	104
16	25	91	116
24	27	103	130
32	28	119	147

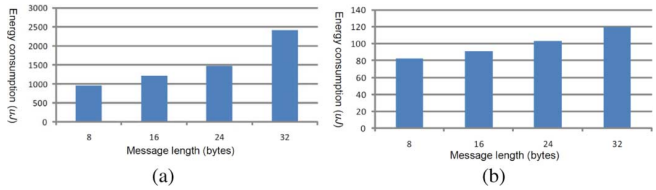
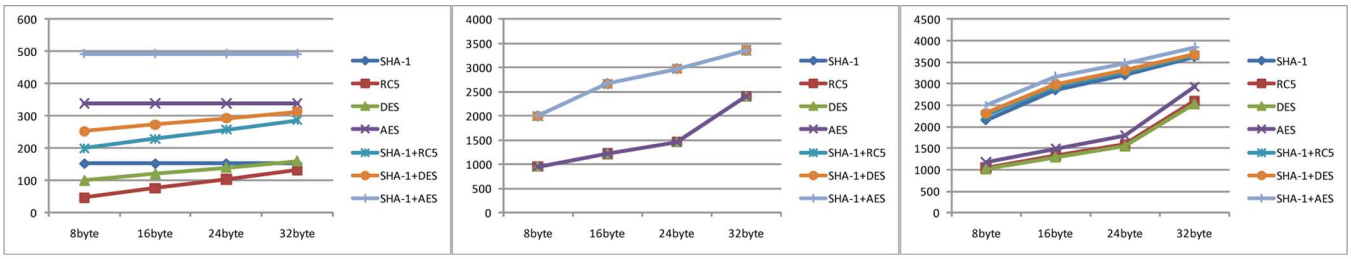


Fig. 3. The energy consumptions on the RF module. (a) CrossBow. (b) Ember.

We measure the energy consumptions of these two modules without integration of security algorithms. We use Tektronix TDS5032B oscilloscope to measure the voltage variation and operating time of the node. After getting the voltage V of a sensitive resistor with resistance R, we can use Ohm's law $I = V/R$ to compute the current value I. Hence, according to the resistance R of WAMS node, we get power $P = I^2 * R$, and energy $E = P * T$, where T represents the time of code execution and data transmission of the node.

And we set these energy consumptions as the baseline consumptions. For messages with four different lengths, we measure the energy consumption when a node computes the data as well as the energy consumption when it sends the data, as shown in Tables II and III. In these tables, the "E_CPU" represents the energy consumption on the CPU, and "E_RF" represents the energy consumption of the RF module. The "E_Sum" represents the sum of these two parts.

The CrossBow and the Ember nodes use the same processors. Therefore, for the same application, the energy consumptions of CPU on these two nodes are the same. The major difference is the physical layer protocol. The CrossBow node uses the 868/915 MHz physical layer standard defined in IEEE 802.15.4, while the Ember uses the 2.4 GHz physical layer standard [24]. There is significant difference in data transmission rate: the CrossBow has the speed of 19.2–38.4 kb/s, while the Ember has the speed of 250 kb/s. As shown in Tables II and III, when sending messages with the same length, the total energy consumption on the Ember is far less than that on the CrossBow. When the length increases, the energy consumption on the CPU does not vary significantly, but the one on the RF module does.



(a) (b) (c)

Fig. 4. The energy consumptions when running different security algorithm on a CrossBow node. (a) On the CPU. (b) On the RF module. (c) On the whole node. In (b), the three curves for RC5, DES, and AES overlap into the lower curve, and the other four are overlapped into the upper curve. In (c), RC5 and DES are overlapped into the lowest curve, above which is the AES curve. SHA-1 + AES is the highest curve, and the left three overlap into the curve just below it.

The energy consumption on the RF module with 4 different message lengths is shown in Fig. 3.

For the Ember node, the energy consumption on the RF module increases linearly as the length of message increases. For the CrossBow node, when the length of the message is between 8 to 24 bytes, the energy consumption increases linearly. However, when the length of message increases from 24 bytes to 32 bytes, the energy consumption has a significant increase, about 64.7%. The reason is that the XMesh protocol implemented in the CrossBow node defines the maximum size of the data packet to be 29 bytes. A 32 bytes data needs to be separated into two packets, leading to more energy consumption. And the maximum size of the data packet on the Ember node is 68 bytes. It can send message with the length from 8 to 32 bytes in just one packet.

A. *Energy Consumption of Security Algorithm on CrossBow*

In this experiment, we measure the energy consumption of the CrossBow node when running different security algorithms on messages with four different lengths. We set the energy consumption of the CrossBow node with no security service as the baseline consumption, and measure the energy consumption of different security algorithms. We also compare the energy consumption on the CPU, the RF module, and the whole node.

The energy consumptions on the CPU when running different security algorithms are shown in Fig. 4(a). For example, when running the SHA-1 algorithm, the energy consumption on the CPU is 153 μJ, 6.9 times the energy consumption when the CPU executes the data processing program. This means that the computational complexity of a security algorithm is far more complex than the one of an application. Since the step length of the SHA-1 algorithm is 64 bytes, the energy consumptions of this algorithm is the same, when the message lengths are between 8 to 32 bytes. Similarly, the step length of the AES algorithm is 128 bytes, the energy consumption with those four different lengths messages are all 339 μJ. On the other hand, the step lengths of both the RC5 and the DES are $8N$ bytes, $(N = 1, 2, \ldots)$. The energy consumptions of these two algorithms increase linearly as the length of message goes up.

For those messages with four different lengths, the energy consumptions on the CPU when executing the RC5, the Data Encryption Standard (DES), and the Advanced Encryption Standard (AES), are 90 μJ, 130 μJ, and 339 μJ on average, respectively. It means that the energy consumption is linearly dependent on the strength of the security algorithm. Secure Hash Algorithm (SHA) was published as federal information processing standard in 1993 and SHA-1 is a revised version. In Fig. 4(a), "SHA-1+RC5," "SHA-1+DES," and "SHA-1+AES" are three different combinations of encryption and integrity protection service [25]. The results show that the energy consumption of a combination is far larger than a single service. This reflects the linear relationship between the energy consumption and the strength of the security service. For the message with a length of 32 bytes, the highest energy consumption on CPU is the one when running the "SHA-1 + AES" security scheme, 493 μJ, 17.6 times higher compared to the baseline energy consumption. However, this part of energy consumption just accounts for 20.2% of the total energy consumption, which is 2443 μJ. Thus, the energy consumption when executing the security algorithm will not impact the total energy consumption of a node at large.

The energy consumptions on the RF module when executing different security algorithms on a CrossBow node are shown in Fig. 4(b). Comparing with Table II, we find out that implementing these three encryptions (RC5, DES, and AES) individually will not increase the energy consumption on the RF module, due to the fact that the lengths of messages do not increase after the encryption. However, the energy consumption increases by 98.0% on average when executing the SHA-1 algorithm. The reason is that the SHA-1 algorithm adds 20 bytes of abstract information, causing additional work load to the RF module. Furthermore, in some cases, the extended message may need multiple packets for transmitting, which requires more energy consumption.

Finally, we measure the energy consumptions of the whole node, when running different security algorithms. The results are shown Fig. 4(c). Compared with the "E_sum" in Table II, running the RC5, the DES, and the AES individually, the energy consumption increases by 6.6%, 4.0%, and 20.2%, respectively. And the Hash algorithm increases the energy consumption significantly, 104.3% on average, with the SHA-1 algorithm on messages with four different lengths. This means that the lifetime of a CrossBow node decreases by half. The three combinations with the Hash algorithm bring 110.2% increase on average. Thus, for an energy-constrained CrossBow node, the use of the Hash algorithm needs careful consideration.

B. *Energy Consumption of Security Algorithm on Ember*

Similar to the experiment in the previous subsection, we conduct the measurement and analysis of the energy consumption of security algorithms on the Ember node. First, we measure the

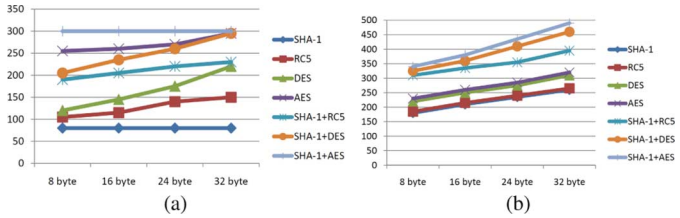


Fig. 5. The energy consumptions when running different security algorithms on a Ember node. (a) On the CPU. (b) On the whole node. In (b), RC5 and SHA-1 overlap into the curve at the lowest, above which is the curve that DES and AES overlap into.

energy consumption on the CPU, as shown in Fig. 5(a). For the four different lengths messages, the energy consumption on the CPU when running SHA-1 is 82 μJ on average. The energy consumptions of RC5 are between 105 μJ to 151 μJ. The encryption algorithm consumes 54% more energy than the Hash algorithm does. And the consumption further increases when the Hash and the encryption algorithms are implemented in the node.

Comparing the results in Fig. 5(a) and the "E_Sum" in Table III, we find out that the energy consumptions on the CPU when executing the DES and other two combination schemes are larger than the total energy consumption of a node when there is no security service on it. Therefore, the encryption algorithm increases the total energy consumption significantly.

We further analyze the energy consumption impact of the security algorithm on the RF module. Without integrating the security service, the energy consumption on the RF module is 82 μJ to 119 μJ, with 8 to 32 bytes messages. Compared to the CPU, the RF module is the major part in the Ember node consuming energy. When the message is processed with the Hash algorithm, the energy consumptions of the RF module are 120 to 159 μJ, on four different lengths messages. The variation of the consumption on Ember is relatively small, due to the transmitting characteristics of the Ember node.

Finally, we measure the total energy consumptions on the Ember node with different security algorithms, as shown in Fig. 5(b). Comparing the results shown in Fig. 5(b) and the "E_Sum" in Table III, we can find that implementing the SHA-1, the RC5, and the DES individually has increased the energy consumption by 77.4%, 81.4%, and 112.1%, respectively. This means that the amount of energy consumed on the Ember node by the Hash and the encryption algorithms does not have big differences. Implementing both these two kinds of algorithms will cause the energy consumption increases by 2.9 times. Thus, the high level security algorithms will decrease the lifetime of the Ember node by 60% on the average.

C. Comparison of the Crossbow and the Ember Node

For the basic computational task, the energy consumptions on the CPU are almost the same on the Ember node and the CrossBow node, as shown in Tables II and III. When running the SHA-1 algorithm, the energy consumption on CPU of an Ember node is only half of that of a CrossBow node, due to the different compilers and the different memory constraints of code porting. For the encryption algorithm, the CPU energy consumptions are similar on these two nodes.

When using the SHA-1 algorithm with the same length messages, the RF module of a CrossBow node consumes 10 times energy as that of an Ember node. The reason is because these two nodes have different data transmitting speed, with a speed ratio of 1:11. Since the power of these two nodes are almost the same (around 50 mW), the required computing time makes the difference.

TABLE IV
THE COMPARISON OF TOTAL ENERGY CONSUMPTIONS

The length of messages (bytes)	E_Sum(μJ)			
	SHA-1+RC5		SHA-1+DES	
	CrossBow	Ember	CrossBow	Ember
8	2260	307	2318	322
16	2942	331	2987	362
24	3278	356	3312	407
32	3657	392	3681	456

To compare the total energy consumption, we choose two combinations of security scheme: (1) "SHA-1 + RC5" and (2) "SHA-1 + DES" and measure the total energy consumptions of them, as shown in Table IV.

Table IV shows that the energy consumption on a CrossBow node is 8.9 times as the one on an Ember node, for the "SHA-1+ RC5" combination. For the other combination, the CrossBow consumes 7.9 times energy as the Ember node does. Thus, for the slow transmitting speed nodes, it is necessary to control the overhead caused by security algorithms, and the computational complexity does not impact the energy consumption significantly. Besides, as shown in Table IV, for shorter messages (8–16 B), those two combinations perform similarly. For longer messages, the "SHA-1 + RC5" combination is less energy-consuming. Thus, the security service selection in energy-constrained system should be based on the result of energy measurements.

V. ENERGY OPTIMIZATION FOR ENCRYPTION ALGORITHMS

A. Inside Each Iteration

In the symmetric-key encryption algorithm, the encryption and decryption algorithms need to process the plaintext or the ciphertext with substitution iteratively [26]. In order to speed up the execution, two optimization techniques, the lookup table and loop unfolding [27], are often used in software implementations. In this section, we will use AES algorithm as an example to study how to reduce the energy consumption through code optimization.

The core of AES algorithm has two parts [26]: secret key scheduling and multi-iteration encryption substitution. The major part of energy consumption is multi-iteration encryption substitution. The pseudocode of multi-iteration encryption substitution is shown in Fig. 6(a).

In every iteration, the four fundamental operations of AES: SubByte, ShiftRow, MixColumns, and AddRoundKey, can be implemented through predefined lookup tables in order to speed up the computation of each iteration [26]. But this will cost more storage space.

Loop unfolding operation will copy the loop body to reduce the number of substitution iterations. The purpose is to increase the parallelism of the instructions and reduce the time for jump and branch operations in the loop body [27]. For example, a one-time loop unfolding will copy the loop body once. This will increase the code size one time but reduce the iteration number by half. Fig. 6(b) shows the result after one-time loop unfolding

```
For round=1 to N {
    SubBytes(state);
    ShiftRows(state);
    MixColumns(state);
    AddRoundKey(state,
      W[round*Nb,(round+1)*Nb-1]);
    }
```

(a)

```
For round=1 to N/2 {
    SubBytes(state);
    ShiftRows(state);
    MixColumns(state);
    AddRoundKey(state,
      W[round*Nb,(round+1)*Nb-1]);
    SubBytes(state);
    ShiftRows(state);
    MixColumns(state);
    AddRoundKey(state,
      W[(round+1)*Nb,(round+2)*Nb-1]);
    }
```

(b)

Fig. 6. (a) Iteration of AES encryption. (b) Iteration of AES encryption after 1 loop unfolding $(f = 1)$.

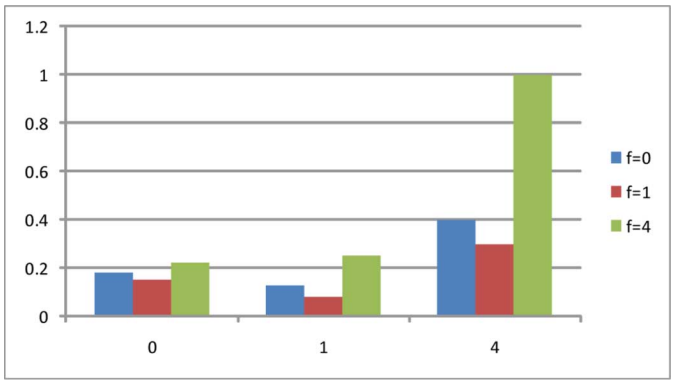


Fig. 7. The impact of the number of lookup tables and the number of unloop on the energy consumption of AES.

$(f = 1)$ for AES. Obviously, we need be careful to use loop unfolding for the WAMS nodes with tight storage constraints because of the storage cost of loop unfolding. We use a series of experiments to explain the impacts of different number of loop unfolding and lookup table operations on the total energy consumptions of CPU.

First, we use different number of lookup table and loop unfolding operations to optimize AES algorithm. Then, we have measured the total energy consumptions of different ways of AES to encipher 64 bytes data. The results are shown in Fig. 7. The vertical axis represents the total energy consumptions after being normalized. The horizontal axis represents the number of lookup tables, i.e., 0, 1, and 4. Three different bars represent the energy consumptions of three different number of loop unfolding times $(f = 0, 1, 4)$.

Below, according to the number of lookups, we will compare the results shown in Fig. 7 and analyze the reasons that cause the change of energy consumptions.

- Without a lookup table, the three different loop unfolding operations have almost the same energy consumptions.
- With only one lookup table to realize SubByte change, the energy consumption for 0 or 1 loop unfolding is reduced 41.5% compared with the scenario without a lookup table. The reasons for this reduction are: Using one lookup table, we can increase the efficiency of implementation and save the energy consumption of CPU. With one loop unfolding, we can further increase the efficiency and reduce the energy consumption of CPU. But with too many loop unfolding (such as four times), the energy consumption will rise sharply because the energy-saving caused by efficiency increase cannot make up the energy penalty caused by frequent visiting of memory.

TABLE V
THE ENERGY CONSUMPTION OF FOUR IMPLEMENTATIONS OF AES

Key size (bits)	ECB $(\mu J/B)$	CBC $(\mu J/B)$	CFB $(\mu J/B)$	OFB $(\mu J/B)$
128	10.1	11.2	11.8	12.2
192	11.3	12.3	12.4	12.8
256	11.9	12.8	13.3	13.7

- When we use four lookup tables to implement the algorithm, the total energy consumptions for all three loop unfolding methods will be sharply increased and have a big difference. Among them, one time loop unfolding has lowest energy consumption, while four-time loop unfolding has 3.1 times of energy consumption compared with one-time loop unfolding. The reason for sharp energy rising is because of the energy penalty caused by efficiency improvement.

B. Configuration of Encryption Algorithms

We will explain how to configure the encryption algorithms through some experiments. First, encryption algorithms have four operation models: ECB, CBC, CFB, and OFB. Among them, ECB is the simplest model, and the most vulnerable for attack model. Other three models have adopted different loop feedback mechanisms and have stronger capabilities to resist to attack. We have measured the energy consumption of AES encryption on 32 bytes data with three different key-length under four different models. The results are shown in Table V. From the experimental results, we can see that the energy consumption by AES under ECB model is the smallest no matter what size of key length while the energy consumption under other three models are similar.

Second, we will analyze the impacts on energy by the number of iterations and secret-key length. With a CrossBow node, we have measured the total energy consumption of RC5 on 32 bytes of data with different key lengths and iteration numbers. The results are shown in Fig. 8. The horizontal axis represents the number of iterations and the vertical axis represents energy (unit: μJ/B). The three bars represent the energy consumptions of RC5 with 56, 128, and 256 bits in key length. From Fig. 8, we can see that with the same key length, the average energy consumption of 16 iterations encryption is 3.3 times of that of 8 iterations. With same number of iterations, the impact of key length on energy is small, especially with large number of iterations. For example, with 16 iterations, the energy consumption with 256 bits key length is only 7.2% higher than that of 56 bits key length.

Third, as we know, the attack on key is a simple and effective way in security attack. Hence, the security of encryption service needs a key in certain length. The increase in iteration number will increase the ability to resist the attack from secret-key analysis. To balance security and energy consumption, we should adjust the iteration number under tight energy scenarios.

Finally, the step length of block cipher has certain impacts on energy consumption. The block ciphers, such as RC5, DES, and AES, transfer the bit stream with 8 or 16 bytes into a pseudorandom series through key. DES and RC5 encrypt with $8N(N = 1, 2, \ldots)$ bytes as step length while AES uses 128 bytes fixed step length.

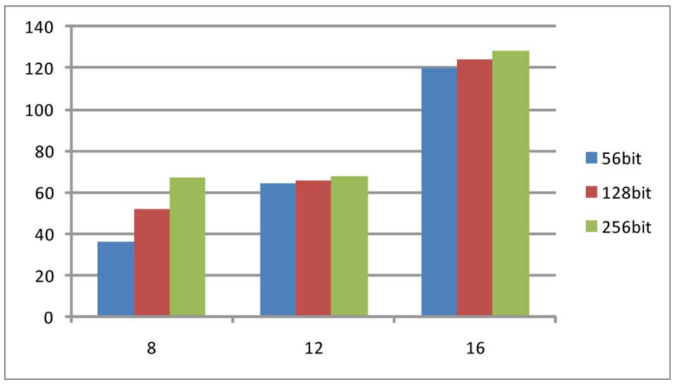


Fig. 8. The impact of the iteration and the key length on the energy consumption.

VI. The Principles for Security Algorithms

In a system with resource and energy constraints, such as a WAMS node, we need to consider the balance of the three factors: energy consumption, security strength, and time. According to the measurement and analysis of security algorithms, we propose the following principles for the application of security algorithms on energy-constrained systems.

1) Principle 1 (Encryption Algorithm Selection Principle): We need balance the three factors: energy, security, and time. Energy consumption is one of the key factors in selecting encryption algorithm if the other two factors (security strength and time) are similar.

We suggest to use encryption service with RC5 algorithm. Comparing with other encryption algorithms, RC5 has the following advantages: 1) According to energy consumption, shown in Figs. 4(a) and 5(a), for different length encryption data, the energy consumption of RC5 is significantly lower than that of AES and DES. 2) The design of RC5 is concise and it does not need a lookup table with large storage. The memory cost of RC5 is significantly smaller than that of AES. 3) We can customize the group size, secret-key length, and the number of iterations of RC5, which can be used flexibly in systems with different resource configurations. 4) RC5 is better than DES in security strength and implementation efficiency.

Furthermore, RC6 is another choice. RC6 can speed up the diffusion through bringing multiplication operations on RC5. Although multiplication will increase the energy consumption of CPU, we can decrease it through reducing the number of encryption iterations. Also, RC6 has higher security strength. Till now no security defect has been found. J. Grobschadal *et al.* [28] has pointed out that RC6 is suitable for memory-constrained systems. Furthermore, we can select TEA encryption algorithm for WAMS nodes with weak computation capabilities.

2) Principle 2 (Intra-Iteration Principle): We can increase the implementation efficiency and reduce energy consumptions through code optimization of encryption algorithms with suitable amounts of loop unfolding and lookup table operations.

For encryption algorithms with multi-iteration, the major part of energy consumption is multi-iteration encryption substitution. From the comparison and analysis in previous section, we can get the above intra-iteration principle. For example, for AES in Fig. 7, we can get good results with only one SubByte lookup table and one loop unfolding.

3) Principle 3 (Algorithm Configuration Principle): The security strength of encryption algorithms is related to operation model, key length and the number of iterations. We can change these parameters to affect the total energy consumption.

According to the experimental results in previous section, we summarize the following detailed rules: 1) For operation models: ECB model has the smallest energy consumption and can be used in tight energy-constrained scenarios. While the other three models have larger energy consumption compared with ECB model and should be used in the case that has sufficient energy to improve the security of a system. 2) Iteration number is the decision factor of energy consumption for symmetric encryption. 3) We suggest to subtly increase the key length and decrease the number of iterations for the energy-constrained systems, such as WAMSs. 4) It will improve the implementation efficiency and reduce energy consumption when the message in processing is set as the same length as the step length of the cipher.

VII. Conclusion

WAMS is one of the most critical parts of the smart power grid. The security of WAMS faces great challenges while satisfying energy constraints. The key point in applying security algorithms to WAMS is to balance the energy consumption and the energy supply of the systems. In this paper, we take power nodes (sites) as platforms to experimentally study ways of energy consumptions in different security algorithms. The paper has two contributions: first, we have proposed an array of novel code optimization methods to increase energy consumption efficiency of different security algorithms; second, the experimental results demonstrate that it is extremely beneficial to select and configure security algorithms in energy-constrained systems in order to improve the security of the systems.

References

[1] J. Ree, V. Centeno, J. Thorp, and A. Phadke, "Synchronized phasor measurement applications in power systems," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 20–27, 2010.

[2] B. Luca and G. D. Micheli, "System-level power optimization: Techniques and tools," *ACM Trans. Design Autom. Electron. Syst.*, vol. 5, no. 2, pp. 115–192, 2000.

[3] M. Qiu, L. T. Yang, Z. Shao, and E. H.-M. Sha, "Dynamic and leakage energy minimization with soft real-time loop scheduling and voltage assignment," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 3, pp. 501–504, Mar. 2010.

[4] M. Qiu and E. H.-M. Sha, "Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems," *ACM Trans. Design Autom. Electron. Syst. (TODAES)*, vol. 14, no. 2, Art. 25, pp. 1–30, Apr. 2009, ACM TODAES 2011 Best Paper Award.

[5] Atemu—Sensor Network Emulator/Simulator/Debugger [Online]. Available: http://www.hynet.umd.edu/research/atemu/

[6] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. 2nd ACM Int. Conf. Embedded Networked Sensor Syst.*, 2004, pp. 188–200.

[7] O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate prediction of power consumption in sensor networks," in *Proc. 2nd IEEE Workshop Embedded Networked Sensors*, 2005, pp. 37–44.

[8] E. Perla, A. Cathain, R. S. Carbajo, M. Huggard, and C. M. Goldrick, "PowerTOSSIM z: Realistic energy modelling for wireless sensor network environments," in *Proc. 3nd ACM Workshop Performance Monitoring Meas. Heterogeneous Wireless Wired Networks*, 2008, pp. 35–42.

[9] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2005, pp. 324–328.

[10] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded Internet," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2005, pp. 247–256.
[11] M. Hadley, J. McBride, T. Edgar, L. O'Neil, and J. Johnson, "Securing wide area measurement systems," Pacific Northwest Natl. Lab., Tech. Rep., 2007.
[12] F. Li, W. Qiao, H. Sun, H. Wan, J. Wang, Y. Xia, Z. Xu, and P. Zhang, "Smart transmission grid: Vision and framework," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 168–177, 2010.
[13] Y. Zhang, P. Markham, T. Xia, L. Chen, Y. Ye, and Z. Wu *et al.*, "Wide-area frequency monitoring network (FNET) architecture and applications," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 159–167, 2010.
[14] A. Armenia and J. Chow, "A flexible phasor data concentrator design leveraging existing software technologies," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 73–80, 2010.
[15] P. Zhang, F. Li, and N. Bhatt, "Next-generation monitoring, analysis, and controlfor the future smart control center," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 186–192, 2010.
[16] J. Dwoskin, D. Xu, J. Huang, M. Chiang, and R. Lee, "Secure key management architecture against sensor-node fabrication attacks," in *IEEE Global Telecommun. Conf.*, 2007, pp. 166–171.
[17] A. Agah and S. Das, "Preventing DoS attacks in wireless sensor networks: A repeated game theory approach," *Int. J. Network Security*, vol. 5, no. 2, pp. 145–153, 2007.
[18] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Proc. IEEE Symp. Security Privacy*, 2005, pp. 49–63.
[19] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasure," *Ad-Hoc Network*, vol. 1, no. 2, pp. 293–315, 2007.
[20] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proc. ACM Conf. Comput. Commun. Security*, 2006, pp. 1–10.
[21] J. Lach, D. Evans, J. McCune, J. Brandon, and L. Hu, "Power-efficient adaptable wireless sensor networks," in *Proc. Int. Conf. Military Aerosp. Programmable Logic Devices*, 2003, pp. 1–8.
[22] C. C. Chang, D. J. Nagel, and S. Muftic, "Balancing security and energy consumption in wireless sensor networks," *Lecture Notes in Computer Science*, vol. 4864, pp. 469–480, 2007.
[23] G. D. Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *IEEE Int. Conf. Wireless Mobile Comput., Networking, Commun.*, 2008, pp. 580–585.
[24] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Performance assessment of a class of cross-layer optimized protocols for geographic routing in WSNs," in *Proc. IEEE Int. Symp. Personal, Indoor, Mobile Radio Commun.*, 2008, pp. 1–5.
[25] M. Passing and F. Dressler, "Experimental performance evaluation of cryptographic algorithms on sensor nodes," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst.*, 2006, pp. 882–887.
[26] A. J. Menezes, V. O. Scott, and A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 2006.
[27] V. Alfred, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2 ed. Reading, MA: Pearson/Addison-Wesley, 2007.
[28] J. Grossschadl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed, "Energy evaluation of software implementations of block ciphers under memory constraints," in *Proc. IEEE/ACM DATE*, 2007, pp. 1–6.

Meikang Qiu (SM'07) received the B.E. and M.E. degrees from Shanghai Jiao Tong University, China. He received the M.S. and Ph.D. degrees in computer science from the University of Texas, Dallas, in 2003 and 2007, respectively.

He has worked at Chinese Helicopter R&D Institute and IBM. Currently, he is an Assistant Professor of ECE at the University of Kentucky, Lexington. He has published more than 120 peer reviewed papers, including 40 journal papers. He also published 3 books and held 2 patents. His research interests include embedded systems, computer security, and wireless sensor networks.

Prof. Qiu has been on various chairs and TPC members for many international conferences. He served as the Program Chair of IEEE EmbeddCom'09 and EM-Com'09. He received the Air Force Summer Faculty Award 2009. He is the recipient of the ACM Transactions on Design Automation Electronic Systems (TODAES) 2011 Best Paper Award. He also won three IEEE/ACM international conferences Best Paper Awards (IEEE EUC'09, IEEE/ACM GreenCom'10, and IEEE CSE'10) and one best paper nomination.

Wenzhong Gao (S'00–M'02–SM'03) received the M.S. and Ph.D. degrees in electrical and computer engineering specializing in electric power engineering from Georgia Institute of Technology, Atlanta, in 1999 and 2002, respectively.

He is currently with the Department of Electrical and Computer Engineering, University of Denver, CO. His current teaching and research interests include renewable energy and distributed generation, smart grid, power-system protection, power-electronics applications in power systems, power-system modeling and simulation, and hybrid electric propulsion systems.

Min Chen (SM'09) received the Ph.D. degree in electrical engineering from South China University of Technology, China, in 2004.

He was a Postdoctoral Fellow at SNU and in the Department of Electrical and Computer Engineering at UBC. He is currently an Assistant Professor in the School of Computer Science and Engineering at Seoul National University (SNU), Korea. He has published more than 120 technical papers.

Dr. Chen received the Best Paper Runner-up Award from QShine 2008. He serves as Editor or Associate Editor for several journals. He was a TPC co-chair for several conferences.

Jian-Wei Niu received the B.S. degree in information science from Zhengzhou Institution of Aeronautical Industry Management, Zhengzhou, China, and the M.S. and Ph.D. degrees in computer application from Beihang University, Beijing, China in 1998 and 2002, respectively.

He is now an Associate Professor at Beihang University. His current research interests include embedded and mobile computing.

Lei Zhang received M.S. and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China.

He was awarded a fellowship from the China Scholarship Council to conduct research at University of Texas, Dallas, as a Visiting Scholar. His research interests include embedded systems, compiler optimization, and hardware/software codesign.