Remote Firmware Execution Control in Computational RFID Systems

Die Wu, *Student Member, IEEE*, Li Lu, *Member, IEEE*, Muhammad Jawad Hussain, and Wenyu Yang

***Abstract*— Computational RFID (CRFID) tags are an emerging class of UHF RFID tags, which are capable of communication, sensing, and computation. However, their computational units are pre-programmed to execute a specific task and cannot change their execution flow during runtime. They require a wired interface to reinstall a new firmware, which strictly limits their use. We address this issue by remotely changing the behavior of CRFIDs by switching their firmware during runtime through commercial RFID reader, without any modification to either CRFID tags or EPC protocol. We present the design, implementation, and evaluation of FirmSwitch–a wireless scheme that equips CRFIDs with the capability of switching amongst pre-installed firmwares during runtime. To achieve this goal, we wirelessly pass the encoded instructions to CRFID tag through RFID reader, which leverages the CRFID tags to switch among firmwares and execute them for intended cycles. For concept validation, three prototype tags belonging to two CRFID categories are evaluated. The results show that FirmSwitch incurs an overall switching delay of 5–35.7 ms, and a minimal energy overhead of 48.7 *n*J–69.3 μJ. The success rate of 95% is achieved at the interrogation range of a half meter with no retransmission.**

***Index Terms*— Computational RFID, firmware flexibility, execution control.**

I. Introduction

UNLIKE conventional passive RFID tags, computational RFID (CRFID) tags feature general-purpose programmable MCU, sensing elements and discrete circuits for communication and power harvesting. Broadly speaking, CRFID tags consist of two major categories: software-defined and chip-based CRFID tags. The MCU in former executes both RFID protocol (e.g. EPC C1G2 [1]) as well as auxiliary operations in software, whereas the latter uses commercial chips to execute RFID protocol and utilizes external MCU to perform computational and sensing tasks. Based on WISP as the pioneering CRFID project [2], [3], many other variants [4]–[6] have been developed on the same design. Such designs not only extend the capabilities of RFID tags for sensing and computation [7]–[10], but also open new opportunities for ubiquitous computing [11]–[14].

A major limitation of current CRFIDs is that they are restricted to execute only a single firmware during runtime. In a typical working cycle, the MCU is programmed with a single firmware for an intended task, once powered up, the MCU will perform the computational or sensing task as per the pre-installed firmware. From a broader perspective, the working scheme of current CRFIDs is analogous to the continuous transition of a state machine within a single state for any firmware, which boots from a predetermined point, performs the task under a fixed execution routine and finally returns to the starting point for next round. To change the functionality of CRFIDs, users have to physically connect the programming tools with each individual tag and reprogram the MCUs. Moreover, the maintenance-free CRFID tags are mostly deployed in hard to reach places to perform the sensing and computing tasks, such as structural monitoring [15], implanted glucose detection [16], in-flight moth monitoring [17] and implanted pacemaker control [18]. For such applications, physically accessing the devices becomes cumbersome, making it infeasible to change the functionality of CRFIDs after deployment.

In order to equip CRFID tags with the capability to execute multiple different operations during runtime, a straightforward way is to combine source codes of all the intended firmwares as a single firmware and program it in the MCU during the installation phase. Such solution, however, is not feasible because of the three reasons: (1) Each individual firmware requires a certain amount of RAM, therefore, scaling the firmware size will increase the RAM requirements which can lead to system failure once RAM is overloaded. (2) CRFIDs are transient powered devices, the execution of the combined firmware is more likely to be terminated by power failure [19], [20]. Consequently, the tag will lose its running state and restart the execution from the beginning of the task. (3) As the memory of each firmware execution is protected, it would be infeasible to perform a self-modification during runtime. In this paper, we present FirmSwitch - a novel scheme to switch between independent firmwares to provide multiple functionalities.

A broader concept of FirmSwitch is depicted in Figure 1 in which CRFID is implemented with a boot-loader and multiple individual firmwares. Upon power-up, the tag starts its operation from boot-loader and communicates with RFID reader while following EPC protocol. In case an instruction to switch a firmware is received, boot-loader reboots to load and execute the target firmware for intended cycles. Once the task is successfully executed or the tag is drained of power, the system resumes from boot-loader.

To realize this idea into a practical solution, the following challenges are addressed by the proposed techniques: (a) In order to load and execute the target firmware, MCU has to reboot to re-initialize the RAM and consequently the

Manuscript received December 4, 2016; accepted January 19, 2017. Date of publication February 6, 2017; date of current version March 22, 2017. This work was supported by the National Natural Science Foundation of China under Grant 61472068 and Grant 61602331. The associate editor coordinating the review of this paper and approving it for publication was Dr. Amitava Chatterjee.

The authors are with the University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: wudie.uestc@gmail.com).

Digital Object Identifier 10.1109/JSEN.2017.2664138

1558-1748 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

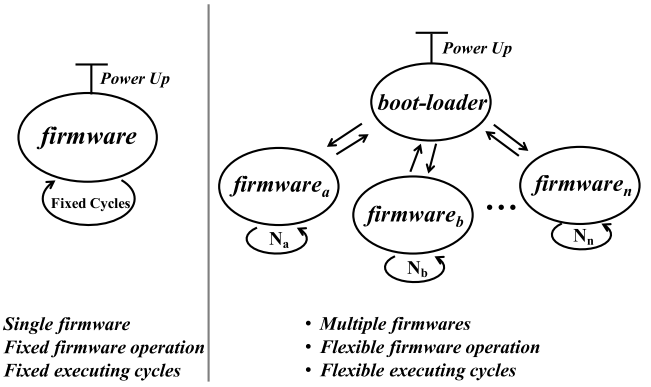


Fig. 1. The firmware execution architecture of current CRFIDs (left) and proposed FirmSwitch scheme (right).

processing state will be lost. To address this issue, the switching parameters are saved in the non-volatile memory before performing the switching operation. (b) CRFID tags work under transient power and the firmware execution might be terminated by the power failure. To comply with transiently harvested power, the MCU of the tag repetitively transits between active and low power modes for energy conservation and system reliability. (c) Real-time CRC computation in CRFID tags might result in a power loss [21]. As a solution, we employ *pre-calculated CRC* for energy efficiency and computational liberty. (d) EPC number of software-defined tag changes each time the data is sent back (as tags embed their processed data in 96-bit EPC field to save power). In result, the reader foresees each 96-bit sensed data as a new tag ID. We address this issue by allocating a unique *pre-defined EPC* to each tag so that reader can recognize CRFID tags during interrogation.

We demonstrate our scheme by implementing a prototype on two types of software-defined tags (WISP5.1, Opt-WISP) and a chip-based tag (Spider). At the system level, we evaluate the time delay incurred during switching to an intended firmware. We also evaluate the energy consumption of the switching operation at different MCU clock frequencies. To validate the system efficiency, we evaluate the success rate of our system at different interrogation ranges.

In summary, we believe that FirmSwitch can considerably reduce the deployment and maintenance overheads for CRFID systems. Following are the key contributions:

- FirmSwitch is a wireless firmware switching approach conforming to EPC protocol so that it requires no modifications to either commodity reader or CRFID tags.
- It equips CRFID tags with execution flexibility whereby the user is able to execute the pre-installed firmwares during system runtime.
- It is light-weight in terms of computation, communication time and power consumption.
- Our system transfers between active and several low power modes so that the switching and execution can still work even if the power failure occurs.

II. Related Work

To best of our knowledge, FirmSwitch is the first approach to offer firmware execution flexibility for CRFID tags through wireless medium using RFID reader and EPC protocol. We only find several relevant works which are specific to MSP430 series of MCUs and tend to manage the firmware execution. Ransford et al. presented an energy aware scheduler [20]. It maps the harvested voltage with appropriate firmware to execute. The selection of new firmware is based upon the extent of energy it utilizes. However, the scheme is restricted only to energy aware scheduling and users does not have the flexibility to select a specific firmware as per the requirement. In Bootie [22], the author presented a pre-programmed look-up table scheduler. The MCU follows a pre-defined look-up table to execute the tasks in a cycle. DewDrop [23] optimizes the efficiency of task execution by means of adapting the execution to the harvested energy. Mementos [24] enables the CRFID tag to complete the long running computations by breaking a single firmware into interruptible executions.

Inspired by WISP [2], many software-defined CRFID platforms have been developed. SoCWISP [25] is a wearable CRFID chip to collect biomedical signals from in-flight insects. Blue Devil WISP [26] adopts a v-shape antenna and improves the performance of power harvesting and communication. EEGWISP [7] is used for collecting EEG signals from the brain. Moo [15] upgrades the MCU for larger memory and collects strain and temperature data from the CRFID tags buried in concrete structures. Moreover, many other designs like WISPs/g [27], WISPCam [28], H-WISP [29], SolarWISP [19] and FrankenWISP [8] are also developed. Besides these software-defined CRFID platforms, we find several commercial chips used in chip-based CRFID tags. These chips include Andy100 [30] by Farsens and SL-series of chips [31] from AMS AG. The former is used to sense data and interact with a mobile robot [32], while the latter is used for temperature and motion detection [33] and soil moisture monitoring [34]. All such platforms will benefit from the firmware flexibility provided by FirmSwitch.

Various approaches have been presented for CRFIDs as far as data transmission, firmware storage and other functionalities are concerned. Harmony [21] pre-calculates the CRC to save harvested energy and execution time. Wisent [35] wirelessly transfers the firmware image to WISP5 through *BlockWrite* command. Authors proposed a scheme [36] to enhance the security of proximity cards with CRFID enabled secret handshakes. Several techniques are presented in Half-wits [37] and FERNS [38] allowing efficient storage and secure data transfer between a CRFID tag and the commercial reader. Several methods aim to improve the data throughput, for example, Flit [39] utilizes the idle slots and transfers short packets in burst, and BLINK [40] leverages the package loss rate and RSSI. Schemes like [41]–[43] implement encryption algorithms on CRFID tags for security. Authors proposed an access control method [44] by making use of electronic and mechanical authentication.

III. FirmSwitch Design Approach

We realize FirmSwitch in form of *Firmware Arrangement* at deployment phase, *Instruction Encoding* on the top of EPC protocol and *Decoding and Execution* on the tag side.

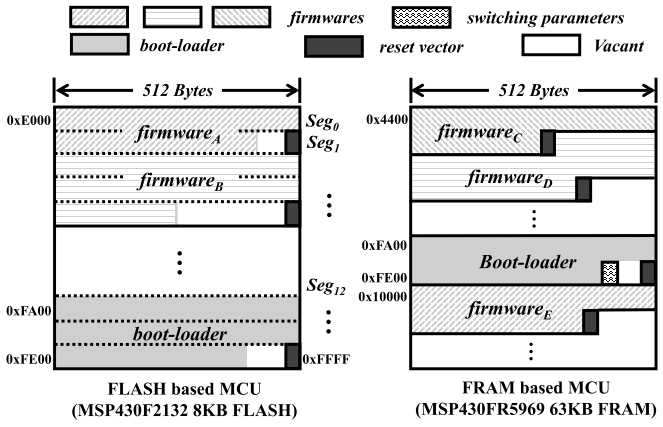


Fig. 2. *Firmware Arrangement* step compiles boot-loader and individual firmware images to on-chip memories. By default, the execution starts from the address contained in 0xFFFE.

The *Firmware Arrangement* step is to allocate tag memory to boot-loader, firmware images and their corresponding reset vectors during deployment phase. At reader side, the user can select a target firmware and specify the intended execution cycles through *Instruction Encoding* step in which the reader embeds the switching parameters inside *Write* command to perform the switching operation. Towards the tag side, the *Decoding and Execution* process communicates with reader while following the EPC protocol, saves the switching parameters when the switching instruction is received, and reboots to load and execute the target firmware for intended cycles based on the parameters parsed from switching instruction.

A. Firmware Arrangement

To make effective use of on-chip memory and provide flexibility for firmware switching and any envisaged wireless reprogramming, the *Firmware Arrangement* step places boot-loader, firmware images, reset vectors in specific regions based on the storage medium of MCU. Current versions of CRFID tags comprise either FLASH based MCU (e.g. MSP430F2132 for Opt-WISP and Spider) or FRAM based MCU (e.g. MSP430FR5969 for WISP5.1). By default, whenever the system is powered-up, the program counter in both FLASH and FRAM based MCUs will point to the reset vector located at 0xFFFE, and the CPU starts execution from the address contained in the reset vector. For both types of MCUs, we leverage this feature for system stability and place the reset vector of boot-loader at 0xFFFE, as shown in Figure 2. As a result, whenever the tag is powered on for the first time or encountered a power outage during execution, it will always start its execution flow from the boot-loader, and system will not be left in unstable stage.

For FLASH-based MCU, the FLASH memory is divided into segments of 512 bytes whereby each segment is also the smallest unit of data erasure [45], [46]. To write FLASH memory, the complete 512-byte segment has to be erased before writing operation. During *Firmware Arrangement*, we stipulate that every firmware should be placed from the beginning of the first memory segment (0xE000). The reset vector of each firmware is stored towards the end of last FLASH segment. Different from FLASH memory, FRAM can be programmed or rewritten in a byte-wise fashion without erasing [47], [48]. As FRAM has no specific memory segmentation, firmwares are placed followed by their reset vectors. Hence, all firmware images (along with their reset vectors) can be stacked one by one starting from address 0x4400 till 0x13FFF.

TABLE I
COMPARISON OF THREE TYPES OF NON-VOLATILE MEMORIES

	FLASH	EEPROM	FRAM
write (2 Bytes)	13.5 ms	6.0 ms	12.8 μs
read (2 Bytes)	63.2 μs	624.2 μs	12.9 μs
writing method	erase(segment)+write	erase(byte)+write	overwrite
endurance	1×10^5	5×10^5	1×10^{14}

When the switching instruction (discussed in Section III-B) is received, the boot-loader will move the relevant reset vector address of the target firmware image to the program counter. In this case, system will reboot to initialize the execution for the target firmware.[1] As a result, switching parameters received from reader will be lost after the system reboot. To address this issue, switching parameters include the reset vector address of target firmware ($Addr$) and execution cycles (N) are saved in the non-volatile memory so that the tag can load them after performing the switching operation. Moreover, as CRFID tags need to frequently update N whenever the execution is complete or a new switching instruction is received, the switching parameters should be saved in the non-volatile memory that incurs the minimum overhead. In CRFID systems, non-volatile memories include on-chip memories like FLASH and FRAM, and off-chip memory like EEPROM. A comparison is shown in Table I. Compared with FLASH, EEPROM performs better as it not only requires lower supply voltage but also incurs less time overhead in updating the parameter. Therefore, for tags with FLASH based MCUs, the switching parameters are saved in the off-chip EEPROM. On the contrary, for tags with FRAM based MCUs, the switching parameters are saved in on-chip FRAM because data accessing speed of FRAM is significantly higher than that of EEPROM. As a result, for FRAM based MCU, a 4-byte memory space is reserved followed by the boot-loader to maintain the switching parameters.

B. Instruction Encoding on Reader Side

In FirmSwitch, we use *Write* command to pass our switching instructions to the tag. Typically, the instruction includes switching indicator, target firmware address and number of execution cycle.

For software-defined CRFID tags, the parameters are embedded in the *MemBank*, *WordPtr* and *Data* field of *Write* command. Unlike commercial tags, the software-defined CRFIDs execute EPC protocol inside it MCU and do not have four separately addressable banks (*User*, *TID*, *EPC*

[1]During initialization, the interrupt vector table, global and static variables, and functions that are purposely designed to be executed from RAM will be copied to the RAM in a sequential order. After that, the program counter points to the entry of the main function and the execution begins.

	Instruction Encoding	*Description*
MemBank (2 Bits)	00 : Routine Operation 11 : Switching Indicator	Distinguish switching operation from routine access command
WordPtr (16 Bits)	0000_h-$FFFF_h$: Target Address	Specify the target firmware to be executed
Data (16 Bits)	0001_h-$FFFF_h$: Execution Cycles 0000_h: Continuous	Specify the intended number of execution

Fig. 3. Switching instruction for software-defined CRFID tags.

Switching Instruction								
•••	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}	T_{n+6}	•••
•••	*FF*	*FF*	*FF*	N_0	N_1	$Addr_0$	$Addr_1$	•••
	Switching Indicator			*# of Executions*		*Target Address*		

Fig. 4. Switching instruction for chip-based CRFID tags.

and *Reserved* banks). Therefore, the parameter saved in the *MemBank* can be used as a flag to distinguish routine *Write* command (coded as 00_b) from switching instruction (11_b) as shown in Figure 3. In case *MemBank*=11_b, the following *WordPtr* stores reset vector address of the target firmware, while *Data* field illustrates the number of intended executions (N). Particularly, *Data*=0000_b indicates that the target firmware should be executed continuously until the power is totally drained.

For chip-based CRFID tags, the commercial RFID chip communicates with external MCU through Serial Peripheral Interface (SPI). However, due to the internal architecture of the commercial RFID chip, it communicates with external MCU only when the *MemBank* field is specified as *User Memory* (i.e. *MemBank* = 11_b). Moreover, at one time, only the least significant 8 bits of *Data* field are transferred to MCU. Therefore, the switching instruction is encoded through the following procedure: we define T_i as the i^{th} data that is transferred from the commercial RFID chip to off-chip MCU. As illustrated in Figure 4, the switching indicator is defined as $T_n \| T_{n+1} \| T_{n+2} = FF_h \| FF_h \| FF_h$ while the number of executions and reset vector address of the target firmware is contained in $T_{n+3} \| T_{n+4}$ and $T_{n+5} \| T_{n+6}$, respectively.

C. Decoding and Execution on Tag Side

Till this step, we have embedded the intended firmware images inside tag's on-chip memory during *Firmware Arrangement* step. Therefore, once the reader encodes the instructions during *Instruction Encoding* step and passes them to the tag, the *Decoding and Execution* process will select the firmware based upon its address, and execute it for the specified cycles. Moreover, we highlight that our system operates among various power saving modes due to the transient power. To further conserve power, for software-defined CRFID tags, we make use of *pre-defined EPC* and *pre-calculated CRC*.

1) Decoding and Execution: Software-defined CRFID tags decode the EPC commands and parse the parameters inside their MCUs. On the other hand, chip-based CRFID tag decodes the instructions inside its chip and transfer information to external MCU through SPI. Therefore, the processes of instruction decoding and execution for software-defined and chip-based CRFIDs are different.

In case of software-defined CRFID tags, the system will initialize with boot-loader and wait for reader's instructions. Once a *Write* command is received, the boot-loader will parse *MemBank*, *WordPtr* and *Data* fields and store them in the variables $Indicator$, $Addr$ and N after validating the CRC. For $Indicator$=00_b, boot-loader continues to run EPC protocol. For $Indicator$=11_b, if the CRFID tag embedded with FRAM based MCU, the boot-loader will save $Addr$ and N into the on-chip FRAM at the specified address (0xFFF0-0xFFF3). Otherwise, if the CRFID tag embedded with FLASH based MCU, the boot-loader will save N to the specific place (first four bytes) in the off-chip EEPROM. Only when the FRAM writing completes or the EEPROM writing succeeds, the target address would be moved to program counter. After that, boot-loader reboots to load the target firmware. Once the firmware is loaded, it reads N and keeps the track of number of successful executions. As a result, N is updated to $N-1$ for each successful execution. After N times of successful executions, the tag will set program counter to 0xFFFE to load the boot-loader and wait for the next instruction.

In case of chip-based CRFID tags, once the SPI communication is initialized, the boot-loader receives the data, labels the i^{th} 8-bit as T_i and saves it in the RAM. Once the switching indicator ($T_n \| T_{n+1} \| T_{n+2} = FF_h \| FF_h \| FF_h$) is received, the boot-loader will write the next two bytes $T_{n+3} \| T_{n+4}$ to the specific address to maintain the number of executions based on the memory type of its MCU, while the following two bytes, ($T_{n+5} \| T_{n+6}$), will be moved to program counter to perform the switching operation.

2) Transition Between Active and Low Power Modes: The MSP430 series of MCUs provide two types of operating modes: Active Mode (AM) and software selectable Low Power Mode (LPM). The Low Power Mode is further categorized in five types: LPM0 to LPM4, which offer various functionalities. In FirmSwitch, the MCU moves between active and low power modes depending upon interrupts generated by the energy supervisor or peripheral devices.

As shown in Figure 5(a), the software-defined CRFID tags power up from LPM4 (Step-1). When sufficient power is harvested and energy supervisor generates an interrupt, the system enters into LPM1 (Step-2) and checks the energy supplied to the MCU. If the power is enough for active mode, the MCU enters AM and gets N from non-volatile memory to check whether the previous firmware execution is completed or not (Step-3). If $N = 0$, system goes into LPM4 and waits for a hardware trigger to indicate the command from the reader (Step-4). Upon receiving the command and having adequate power, the system moves in AM to process the instructions following EPC protocol (Step-5). System moves back to LPM4 to wait for command from reader if it does not receive the switching instruction (Step-4). If the switching instruction is received in step-5, system will go through

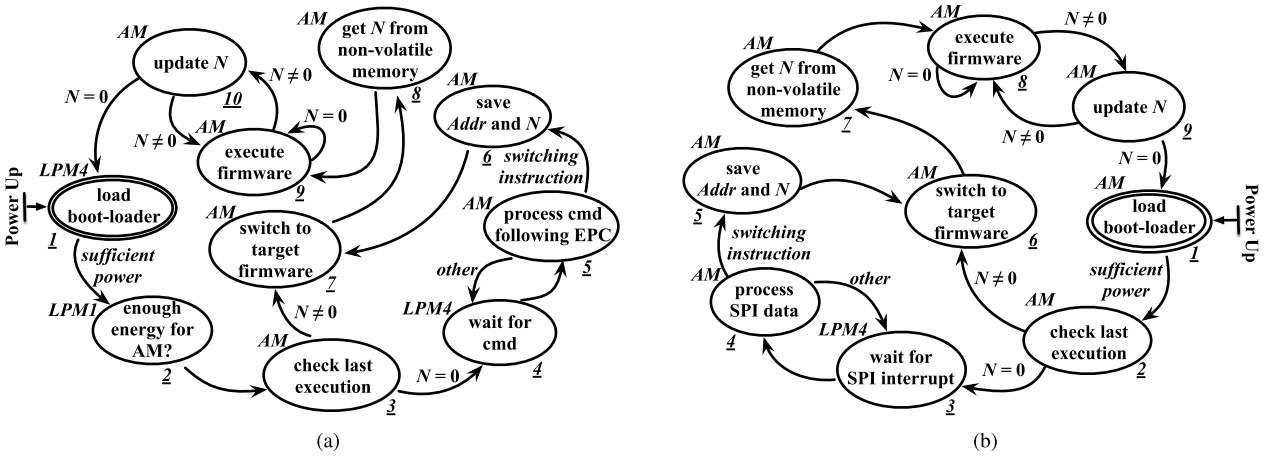


Fig. 5. Illustration of FirmSwitch transition between Active Mode and various Low Power Modes. (a) Transition in software-defined CRFIDs. (b) Transition in chip-based CRFIDs.

Step-6 and Step-7 to save the switching parameters $Addr$ and N in the non-volatile memory and reboot to switch to the target firmware. Then, once the target firmware is loaded into the RAM, it goes into AM to get N from non-volatile memory (Step-8) and execute the firmware (Step-9). In the event where $N = 0$, the execution cycle in the switching instruction is zero (continuous mode), firmware executes continuously until energy drains. Otherwise, upon completion, system updates N in Step-10 and backs to Step-9 if $N \neq 0$. After executing for intended cycles, system returns to Step-1 to initialize boot-loader and moves in LPM4. If $N \neq 0$ in Step-3, system moves to Step-7 to switch to last target firmware and executes it based on the $Addr$ and N stored in the non-volatile memory. For the case once the harvested power is completely drained during the transition, system will restart from Step-1, same as power-on reset.

For chip-based CRFID tags, once sufficient power is harvested, the CRFID chip executes EPC protocol and supplies auxiliary power to the external MCU. As shown in Figure 5(b), once the off-chip MCU is powered up, it loads boot-loader and checks the last execution status in AM (Step-1 and Step-2). If the last execution is completed, system moves to LPM4 and waits for SPI interrupt (Step-3). When the RFID chip initializes the SPI communication and transfers data to MCU through SPI, the MCU moves to AM to process the SPI data (Step-4). If the switching instruction is received, MCU stays in AM and saves the reset vector address of target firmware and execution cycles in the non-volatile memory (Step-5). Then, MCU moves to Step-6 to switch to the target firmware. After loading the target firmware into RAM, MCU gets N from the non-volatile memory (Step-7). When $N = 0$, MCU executes the firmware (Step-8) continuously until power finishes. Otherwise, MCU updates N to $N - 1$ (Step-9) after each successful execution (Step-8). Once N decreased to 0 in Step-9, MCU returns to Step-1 to load the boot-loader. Similar to software-defined CRFID tags, if MCU observed $N \neq 0$ in Step-2, which indicates the last firmware execution is not completed, system will move to Step-6 to switch to the target firmware and execute for the rest number of cycles. In case the power is totally drained, the system restarts from Step-1.

3) Pre-Defined EPC and Pre-Calculated CRC: Commercial RFID tags backscatter their 96-bit unique and reserved EPC number or TID during *Inventory Round*. In contrast, current software-defined CRFID tags embed their sensor data within 96-bit of EPC field, and communicate it during *Inventory Round* of EPC protocol instead of *Access Round* (using Read command). This way, the Tag-ID (EPC number) changes each time the tag replies the sensed or computed data. As a result, the RFID reader foresees each new data (EPC field) as a new RFID tag. We resolve this issue by allocating a unique pre-defined EPC value for each software-defined CRFID token. Normally, current software-defined CRFID tags embed the sensor data in the 96-bit EPC field using a pre-determined format. The 96-bit EPC field comprises the sensor type ID (8 bits), sensor data (64 bits), the hardware version (8 bits) and the hardware serial number (16 bits). For example, the sensor type ID is 0x0F for temperature data and 0x0B for acceleration data. In our approach, to differentiate the pre-defined EPC from the sensor data, an 8-bit unique header (0xFF) that is different from the sensor type ID is utilized at the beginning of each pre-defined EPC number. The tag replies its unique pre-defined EPC number (e.g. $FFFF000000000000000FFFF_h$ in Figure 6) to indicate that it is free to receive any instruction from the reader, i.e., the boot-loader is functional. Therefore, the reader will only send the switching instructions once the pre-defined EPC number is received. Otherwise, a changing EPC number indicates that the tag is executing some firmware and backscattering the sensed or computed data.

During *Inventory Round* and *Access Round* in EPC protocol, another caveat arises once the software-defined tags are required to compute the CRC value during runtime. In EPC protocol, the CRC computation is a mandatory operation in reply to *ACK*, *Req_RN* and access command such as *Write*, *Read* and *Lock*. For chip-based CRFID tags, the CRC computation is performed by commercial chip instead of the external MCU. On the contrary, software-defined tag has to perform the CRC computation inside its MCU. Because of low computational capability and power constraints, computing CRC in MCU during runtime would incur a considerable overhead in both time and energy. In the worst case, the CRC computation might result in a power failure. We resolve this

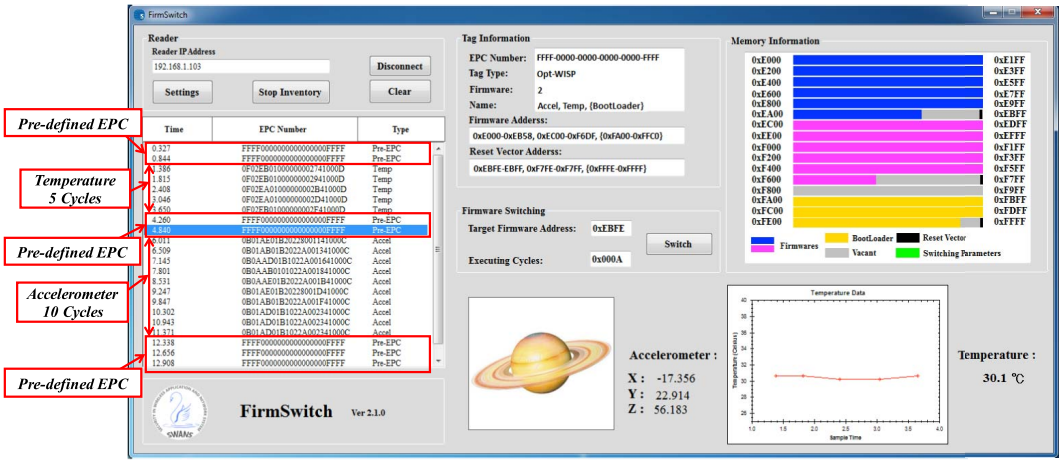


Fig. 6. User Interface passes switching parameters to CRFID tag through RFID reader, receives the reply and displays the results.

issue by employing *pre-calculated CRC*. As the name suggests, the CRC is calculated beforehand, i.e., while programming the tag before its deployment. Following this approach, we pre-compute the CRC values for pre-defined EPC, *Handle* and *Success* and store them in a list which will be used to reply *ACK*, *Req_RN* and *Write* commands, respectively. The tag will select respective CRC value from this list and append it in the reply to the command.

IV. Implementation and Evaluation Metrics

We implement and evaluate FirmSwitch on two software-defined CRFIDs (WISP5.1, Opt-WISP) and a chip-based CRFID (Spider). WISP5.1 is embedded with MSP430FR5969 while the Opt-WISP includes MSP430F2132. The Opt-WISP is designed based on WISP4.1 with additional antennas and harvesters. In case of Spider tag, it uses a commercial CRFID chip (ANDY100) to execute EPC protocol, and MSP430F2132 is interfaced as its external MCU. For evaluation, we develop a user interface for Impinj Speedway R420 RFID reader.

A. User Interface

The user interface is shown in Figure 6, it connects to commodity RFID reader and passes switching parameters to CRFID tags. It also receives the backscattered CRFID tag's and displays the EPC number in the left window. The requisite information of the selected tag including EPC number, tag type, and existing firmwares with its name and address, appears in the *Tag Information* panel. To give a better illustration of the firmware arrangement, a graphic memory map that marks the boot-loader, firmware images and their reset vectors is shown in *Memory Information* panel on the right side. Users can pass the switching parameters from host PC to RFID reader through *Firmware Switching* panel.

User Interface also illustrates the pre-defined EPC and the data sensed by the tag. Once powered up, the boot-loader is loaded by default. In this case, the pre-defined EPC value is visible at 0.327 and 0.844 sec. We then instruct the CRFID tag to switch to the temperature sensing firmware and execute it for 5 times. Next five subsequent changing EPC values (from 1.386 to 3.650 sec) give us the sensed temperature values. Upon completion, the CRFID tag loads the boot-loader again and backscatters the pre-defined EPC number (4.260 and 4.840 sec). Following the same procedure, we instruct the CRFID tag to switch to the acceleration sensing firmware for 10 cycles (6.011 to 13.371 sec), as annotated in the figure.

B. Evaluation Metrics

We evaluate our system at an outdoor place and the evaluation is aimed for switching time, energy consumption and success rate in line with interrogation range.

- *Overall System Delay:* The overall system delay incurred by FirmSwitch. The evaluation includes the time consumption caused by the uplink and downlink communication. This also checks the conformance of FirmSwitch to EPC protocol with commercial RFID reader.
- *Energy Overhead of Switching Operation:* The energy overhead is measured once FirmSwitch switches from boot-loader to another firmware. The evaluation is performed for three clock frequencies: 1, 8 and 16 *MHz*. It excludes the overhead caused by EPC protocol and is specific to the switching operation (setting program counter and saving switching parameters) at different MCU clock frequencies.
- *Success Rate and Interrogation Range:* We calculate the success rate of switching operation in line with interrogation range for three tags, and for comparison, we also our results with EPC transmission [21].

V. Evaluation Results

A. Overall System Delay

In this evaluation, we check the conformity of our scheme with EPC protocol and use commercial RFID reader for experimentation. As shown in Figure 7, the overall system delay is evaluated in the university campus while the distance between CRFID and reader antenna is kept at 0.5 meters. We control Impinj Speedway R420 RFID reader through User Interface, use a circularly polarized 6 dBi antenna and set the output power to 30 dBm. The working frequency of three tags is set to 1 *MHz* and each tag is programmed to switch to execute a LED firmware, i.e., the tag starts from boot-loader,

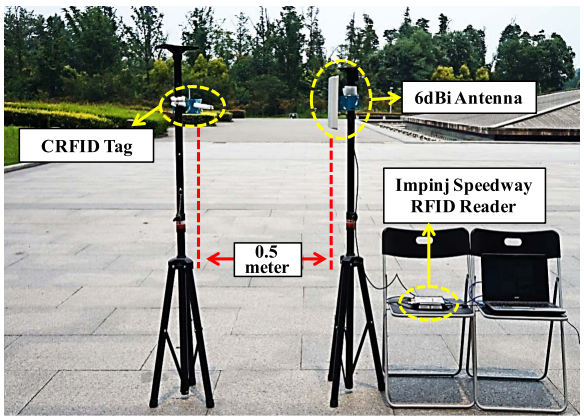


Fig. 7. Experimental setup for overall system evaluation.

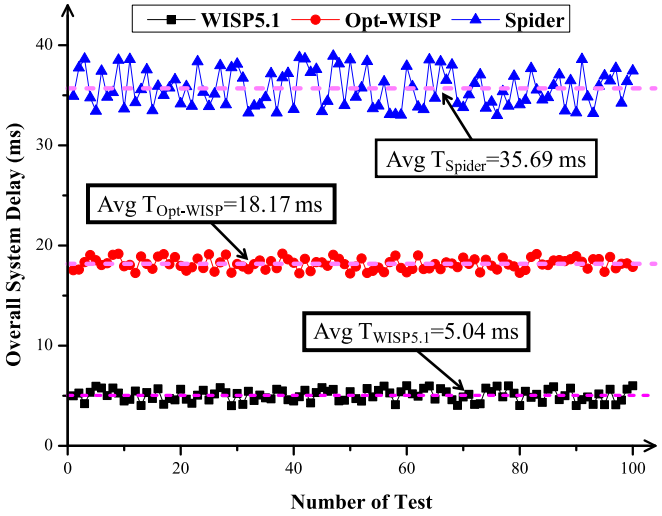


Fig. 8. Overall system delay incurred by FirmSwitch on three platforms.

loads a LED firmware and light on the LED. The time delay is measured from the start of the first command send by the reader till the LED is turned on. The experiment is performed for 100 times and the results are shown in Figure 8. The average time delay of WISP5.1, Opt-WISP and Spider tags is 5.04, 18.17 and 35.69 *ms* respectively as annotated.

We observed that the delay for chip-based CRFID tag (Spider) is significantly higher than software-defined tags (WISP5.1 and Opt-WISP). This is because reader has to send multiple *Write* commands to chip-based CRFID tag for switching from boot-loader to the target firmware. We also observed that the tags equipped with FLASH-based MCU (Opt-WISP and Spider) consume more time compared with the tag with FRAM-based MCU (WISP5.1). The reason is that FLASH-based MCU saves the switching parameters (*Addr* and *N*) into external EEPROM, which consumes much more time for reading and writing compared with FRAM-based MCUs which saves the parameters in FRAM. However, we find that the overall system delay introduced by FirmSwitch is acceptable and our system is fully compatible with EPC protocol and commercial RFID reader.

B. Energy Overhead of Switching Operation

As the energy consumption is a critical parameter for power harvesting designs. The situation for CRFID tags is more serious as a commercial tag typically operates at 150 μW power whereas only the MCU in WISP-CRFID consumes power in amounts of 960 μW [49]. Therefore, we evaluate energy overhead of switching operation on all three platforms. Due to the different working schemes or MCU architectures, the typical firmware that can work on one platform usually cannot work on the other. For ease of comparison, three platform-independent firmware demos [50] are used to evaluate the time and energy overhead. The evaluated firmwares include RC-5 encryption, CRC-16 calculation, and LED blinking Demos with firmware size of 4.5, 2.2, and 0.8 KBytes, respectively. We instruct each tag to switch to one firmware at various clock frequencies (1, 8 and 16 MHz). For measurement, we follow the method of measuring the power consumption of WISP4.1 [51] and put a series resistor of 33 Ω in the power path of the MCU to measure the voltage drop across the resistor, as shown in Figure 9. The voltage before and after the resistor is denoted as V_{before} and V_{after}. We use MCU's GPIO P3.0 as a flag and view its output as a voltage wave on the oscilloscope. The toggling of flag before and after the switching operation indicates that the switching operation has started and completed. The energy overhead is given as

$$Energy\ Overhead = \frac{V_{after} \cdot (V_{before} - V_{after})}{R} \cdot T, \quad (1)$$

where R is the 33 Ω resistor while T represents the time consumption of the switching operation.

TABLE II
TIME AND ENERGY MEASUREMENT FOR FRAM-BASED MCU

Platform	*Target Image*	*Clock (MHz)*	V_{before} *(V)*	V_{after} *(V)*	*Time (μs)*	*Energy (nJ)*
WISP5.1	RC5	1	2.2162	2.1724	1872	5397.7
		8	2.2158	2.1243	237	1395.9
		16	2.2163	2.1035	118	848.4
	CRC	1	2.2157	2.1722	636	1821.1
		8	2.2159	2.1251	80	467.8
		16	2.2161	2.1028	41	296.0
	LED	1	2.2162	2.1746	106	290.6
		8	2.2162	2.1235	13.2	78.7
		16	2.2163	2.1021	6.7	48.7

The energy consumption of switching operation for tag with FRAM based MCU is shown in Table II. The results show a maximum energy overhead of 5397.7 nJ for switching to RC-5 encryption Demo firmware at 1 MHz, and a minimum energy overhead of 48.7 nJ for switching to LED blinking Demo at 16 MHz. We observe that lower MCU clock consumes more power than higher clock frequencies. The relationship between energy and system frequency is quite understandable. As frequency is increased from 1 to 16 *MHz*, the number of computations per second also increases, which in result, reduces the operation delay. Since all three firmwares differ in number of variables and execution flow, the time delay as well as energy consumption varies for each. Moreover, based on the results of three target image at same clock frequency, we found that the increase in the size of the target firmware image results in the increase in time and energy consumption.

For tags with FLASH based MCU, the switching parameters are saved in the off-chip EEPROM. However, in practice,

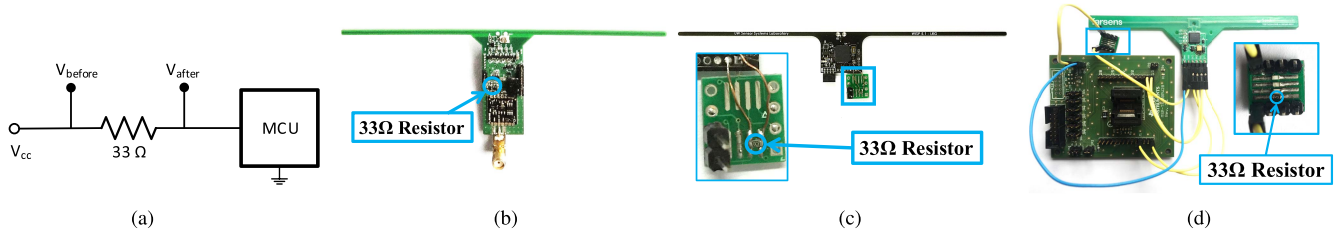


Fig. 9. Experimental setup for energy measurements on Opt-WISP, WISP5.1 and Spider. (a) Circuit Diagram. (b) Opt-WISP. (c) WISP5.1. (d) Spider.

TABLE III
TIME AND ENERGY MEASUREMENTS FOR FLASH-BASED MCU

Platform	*Target Image*	*Clock (MHz)*	V_{before} *(V)*	V_{after} *(V)*	*Time* (μs)	*Energy* (μJ)
Opt-WISP	RC5	1	3.3459	3.2636	8075	65.724
		8	3.3462	3.2598	6447	55.024
		16	3.3461	3.2583	6331	54.884
	CRC	1	3.3458	3.2641	6845	55.315
		8	3.3457	3.2593	6294	53.709
		16	3.3455	3.2587	6255	53.614
	LED	1	3.3467	3.2628	6246	51.813
		8	3.3464	3.2622	6218	51.756
		16	3.3462	3.2623	6215	51.548
Spider	RC5	1	3.1638	3.0717	8078	69.251
		8	3.1641	3.0693	6445	56.827
		16	3.1639	3.0682	6332	56.341
	CRC	1	3.1644	3.0719	6848	58.966
		8	3.1643	3.0681	6296	56.311
		16	3.1642	3.0678	6253	56.037
	LED	1	3.1637	3.0721	6247	53.271
		8	3.1638	3.0719	6219	53.202
		16	3.1636	3.0717	6216	53.173

the off-chip EEPROM should operate up to its own maximum clock frequency. For example, the maximum clock frequency of Microchip 24AA08 [52] used in our system is 400 kHz when the supplied power (V_{after}) is above 2.5 V, and 100 kHz when V_{after} is lower than 2.5 V. Therefore, to reliably and quickly save and get the switching parameters, the EEPROM is operating in a fixed clock frequency (400 kHz) with V_{after} for both Opt-WISP and Spider are higher than 2.5 V. The time and energy measurements for FLASH based MCU are shown in Table III. The *Clock* in the table indicates the working frequency of other MCU operations such as decoding switching instruction, setting the program counter and loading the target firmware image. The results show that, for a certain target image, the energy consumption are similar at certain clock frequencies.

Comparing the results of FLASH based MCU and FRAM based MCU, we found that the former consumes more time and energy in switching to a certain target image at same clock frequency than the latter. Moreover, because of different MCU memories, the energy consumption in WISP5.1 and Opt-WISP differs even though both tags belong to the same family of software-defined CRFIDs. In addition, we observe that the energy overhead is higher when the working frequency is lower for all three platforms.

C. Success Rate and Interrogation Range

In our scheme, the switching instruction is embedded in the *Write* command and sent through commercial reader. However, as CRFID tags are transient powered devices, the harvested power might drain before the switching operation is performed successfully. To validate the performance of our scheme, we evaluate the success rate of the switching operation on WISP5.1, Opt-WISP and Spider at different interrogation ranges.

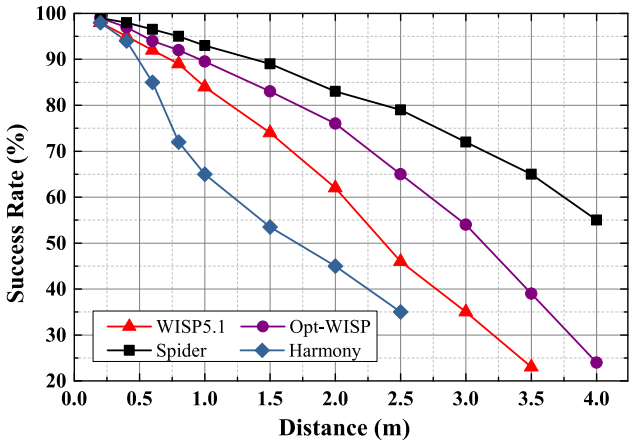


Fig. 10. Success rates at different interrogation ranges.

The success rate is evaluated at an open playground and the experimental setup is similar to Figure 7. In this evaluation, we use the LED blinking Demo as the target firmware for ease of observation. During the experiment, the success rate is evaluated for 200 independent switching instructions with no retransmission for each distance. For comparison, we further evaluate our system against Harmony [21] which uses WISP4.1DL and executes the EPC protocol till *Access Round* to backscatter the sensor data with pre-calculated CRC.

The results are shown in Figure 10. Our approach achieves a success rate of 95% at 0.5 meter on all three platforms. With a more practical interrogation distance of 2 meters, the success rates are still higher than 60%. In practice, RFID reader can retransmit the switching instruction if the instruction is not received and decoded correctly. Compared with Harmony, WISP5.1, Opt-WISP and Spider tags show better success rates and interrogation range. The success rate of Harmony falls sharply after 0.5 meter and the maximum interrogation range is 2.5 meters, while the success rate of WISP5.1, Opt-WISP and Spider decreases gradually and the maximum interrogation range is over 3.5 meters.

Our scheme competes with Harmony because of following reasons: For WISP5.1, an optimized power harvester is used which increases the interrogation range and success rate. For Opt-WISP, the additional power harvester is used exclusively

for computation, therefore, the interrogation range and success rate outperforms WISP5.1. For Spider tag, since the EPC protocol is executed by a commercial chip which offers optimal tag operation, the interrogation range and success rate are the highest among all three platforms.

VI. Conclusion

We present the design, implementation and evaluation of FirmSwitch, a scheme enabling wirelessly switching the firmwares on CRFIDs and executing them for intended cycles. The user compiles all firmwares in a once-for-all fashion during the deployment phase which can be switched back and forth during system runtime through commercial RFID reader and the EPC protocol. To this end, an in-depth discussion is presented for three phases of FirmSwitch: *Firmware Arrangement*, *Instruction Encoding*, and *Decoding and Execution*. We implement FirmSwitch on WISP5.1, Opt-WISP and Spider CRFID tags and evaluate the performance of our scheme in terms of switching time, energy cost and interrogation range. The evaluation results show that FirmSwitch is a viable and practical approach for firmware flexibility in CRFID systems without any modifications to CRFID tag, commodity RFID reader or the EPC protocol.

In future, the successful interrogation range of our approach can be optimized by improving the energy harvesting efficiency of the CRFID tag and reducing power consumption in the tag's physical structure. Moreover, we consider our work to be an elementary step towards "over-the-air programming". Once a CRFID can reliably receive a firmware image and rewrite the MCU, this work can be extended to wirelessly receive, load and execute new firmwares without any wired access.

References

[1] EPCglobal. (Jan. 2005). *EPC Radio-Frequency Identity Protocols, Class 1 Generation 2 UHF Air Interface Protocol Standard Version 1.0.9*. [Online]. Available: http://ams.com/eng/Products/Wireless-Connectivity/Sensor-Tags-Interfaces/SL900A

[2] M. Philipose, J. R. Smith, B. Jiang, A. Mamishev, S. Roy, and K. Sundara-Rajan, "Battery-free wireless identification and sensing," *IEEE Pervas. Comput.*, vol. 4, no. 1, pp. 37–45, Jan. 2005.

[3] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an RFID-based battery-free programmable sensing platform," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 11, pp. 2608–2615, Nov. 2008.

[4] M. Marroncelli, D. Trinchero, V. Lakafosis, and M. M. Tentzeris, "Concealable, low-cost paper-printed antennas for WISP-based RFIDs," in *Proc. IEEE Int. Conf. (RFID)*, Apr. 2011, pp. 6–10.

[5] T. Unander, J. Siden, and H.-E. Nilsson, "Designing of RFID-based sensor solution for packaging surveillance applications," *IEEE Sensors J.*, vol. 11, no. 11, pp. 3009–3018, Nov. 2011.

[6] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall, "Recognizing daily activities with RFID-based sensors," in *Proc. 11th Int. Conf. Ubiquitous Comput.*, 2009, pp. 51–60.

[7] A. Dementyev and J. R. Smith, "A wearable UHF RFID-based EEG system," in *Proc. IEEE Int. Conf. (RFID)*, May 2013, pp. 1–7.

[8] E. Hoque, R. F. Dickerson, and J. A. Stankovic, "Monitoring body positions and movements during sleep using WISPs," in *Proc. Wireless Health*, Oct. 2010, pp. 44–53.

[9] N. Saxena and J. Voris, "Still and silent: Motion detection for enhanced RFID security and privacy without changing the usage model," in *Proc. Int. Workshop Radio Freq. Identificat. Secur. Privacy Issues*, 2010, pp. 2–21.

[10] A. Wickramasinghe and D. C. Ranasinghe, "Ambulatory monitoring using passive computational RFID sensors," *IEEE Sensors J.*, vol. 15, no. 10, pp. 5859–5869, Oct. 2015.

[11] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 39–50, 2013.

[12] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith, "Turbocharging ambient backscatter communication," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 619–630, 2014.

[13] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall, "Wi-Fi backscatter: Internet connectivity for RF-powered devices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 607–618, 2015.

[14] A. N. Parks and J. R. Smith, "Sifting through the airwaves: Efficient and scalable multiband RF harvesting," in *Proc. IEEE Int. Conf. RFID (IEEE RFID)*, Apr. 2014, pp. 74–81.

[15] H. Zhang, J. Gummeson, B. Ransford, and K. Fu, "Moo: A batteryless computational RFID and sensing platform," Univ. Massachusetts Amherst Comput. Sci., Amherst, MA, USA, Tech. Rep. UM-CS-2011-020, 2011.

[16] Z. Xiao *et al.*, "An implantable RFID sensor tag toward continuous glucose monitoring," *IEEE J. Biomed. Health Inf.*, vol. 19, no. 3, pp. 910–919, May 2015.

[17] D. Yeager, F. Zhang, A. Zarrasvand, N. T. George, T. Daniel, and B. P. Otis, "A 9 μA, addressable Gen2 sensor tag for biosignal acquisition," *IEEE J. Solid-State Circuits*, vol. 45, no. 10, pp. 2198–2209, Oct. 2010.

[18] D. Halperin *et al.*, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *Proc. IEEE Symp. Secur. Privacy*, May 2008, pp. 129–142.

[19] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan, "On the limits of effective hybrid micro-energy harvesting on mobile CRFID sensors," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 195–208.

[20] B. Ransford, S. S. Clark, M. Salajegheh, and K. Fu, "Getting things done on computational RFIDs with energy-aware checkpointing and voltage-aware scheduling," in *Proc. HotPower*, vol. 8, 2008, p. 5.

[21] Y. Zheng and M. Li, "Read bulk data from computational RFIDs," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2014, pp. 495–503.

[22] B. Ransford, "A rudimentary bootloader for computational RFIDs," Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep. UM-CS-2010-061, 2010.

[23] M. Buettner, B. Greenstein, and D. Wetherall, "Dewdrop: An energy-aware runtime for computational RFID," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement.*, 2011, pp. 197–210.

[24] B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," *ACM Sigplan Notices*, vol. 47, no. 4, pp. 159–170, 2012.

[25] B. Otis and D. Yeager, "SoCWISP: Ultra-low power wireless sensing RFID chip," in *Proc. WISP Summit Workshop*, 2009.

[26] M. Reynolds and S. Thomas, "The blue devil WISP: Expanding the frontiers of the passive RFID physical layer," in *Proc. WISP Summit Workshop*, 2009, pp. 1–19.

[27] F. Gasco, P. Feraboli, J. Braun, J. Smith, P. Stickler, and L. DeOto, "Wireless strain measurement for structural testing and health monitoring of carbon fiber composites," *Compos. A, Appl. Sci. Manuf.*, vol. 42, no. 9, pp. 1263–1274, 2011.

[28] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith, "WISPcam: A battery-free RFID camera," in *Proc. IEEE Int. Conf. (RFID)*, Apr. 2015, pp. 166–173.

[29] Y. Dong, A. Wickramasinghe, H. Xue, S.-F. Al-Sarawi, and D. C. Ranasinghe, "A novel hybrid powered RFID sensor tag," in *Proc. IEEE Int. Conf. (RFID)*, Apr. 2015, pp. 55–62.

[30] Farsens. (Sep. 2015). *ANDY100 Evaluation Board With Integrated Start-Up Circuit*. [Online]. Available: http://www.farsens.com/media/document/26/ds-spider-h254-v01.pdf

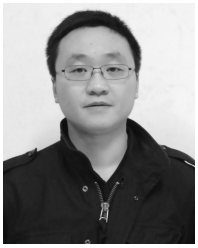
[31] (May 2014). *SL900A, EPC Class 3 Sensory Tag Chip—For Automatic Data Logging*. [Online]. Available: http://ams.com/eng/Products/Wireless-Connectivity/Sensor-Tags-Interfaces/SL900A

[32] J. Wang, E. Schluntz, B. Otis, and T. Deyle. "A new vision for smart objects and the Internet of things: Mobile robots and long-range UHF RFID sensor tags." [Online]. Available: https://arxiv.org/abs/1507.02373

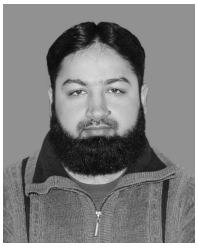
[33] A. Vena, B. Sorli, A. Foucaran, and Y. Belaizi, "A RFID-enabled sensor platform for pervasive monitoring," in *Proc. 9th Int. Symp. Reconfigurable Commun.-Centric Syst.-Chip (ReCoSoC)*, May 2014, pp. 1–4.

[34] C. Bauer-Reich *et al.*, "An investigation of the viability of UHF RFID for subsurface soil sensors," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, Jun. 2014, pp. 577–580.
[35] J. Tan, "Robust downstream communication and storage for computational RFIDs," Ph.D. dissertation, Dept. Softw. Technol., TU Delft, Delft Univ. Technol., Delft, Netherlands, 2015.
[36] A. Czeskis, K. Koscher, J. R. Smith, and T. Kohno, "RFIDs and secret handshakes: Defending against ghost-and-leech attacks and unauthorized reads with context-aware communications," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, 2008, pp. 479–490.
[37] M. Salajegheh, Y. Wang, A. A. Jiang, E. Learned-Miller, and K. Fu, "Half-Wits: Software techniques for low-voltage probabilistic storage on microcontrollers with NOR flash memory," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2, 2013, Art. no. 91.
[38] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.
[39] J. Gummeson, P. Zhang, and D. Ganesan, "Flit: A bulk transmission protocol for RFID-scale sensors," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 71–84.
[40] P. Zhang, J. Gummeson, and D. Ganesan, "Blink: A high throughput link layer for backscatter communication," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 99–112.
[41] C. Pendl, M. Pelnar, and M. Hutter, "Elliptic curve cryptography on the WISP UHF RFID tag," in *RFID. Security and Privacy*. Berlin, Germany: Springer, 2012, pp. 32–47.
[42] A. Szekely, M. Höfler, R. Stögbuchner, and M. Aigner, "Security enhanced WISPs: Implementation challenges," in *Wirelessly Powered Sensor Networks and Computational RFID*. New York, NY, USA: Springer, 2013, pp. 189–204.
[43] H. J. Chae, M. Salajegheh, D. J. Yeager, J. R. Smith, and K. Fu, "Maximalist cryptography and computation on the WISP UHF RFID tag," in *Wirelessly Powered Sensor Networks and Computational RFID*. New York, NY, USA: Springer, 2013, pp. 175–187.
[44] Y. Shu, Y. J. Gu, and J. Chen, "Dynamic authentication with sensory information for the access control systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 427–436, Feb. 2014.
[45] Texas Instrument. (Jan. 2012). *Mixed-Signal Microcontroller*. [Online]. Available: http://www.ti.com/lit/ds/symlink/msp430f2132.pdf
[46] Texas Instrument. (Jul. 2013). *MSP430x2xx Family User's Guide*. [Online]. Available: http://www.ti.com/lit/ug/slau144j/slau144j.pdf
[47] Texas Instrument. (Mar. 2015). *MSP430FR59xx Mixed-Signal Microcontrollers*. [Online]. Available: http://www.ti.com/lit/ds/symlink/msp430fr5969.pdf
[48] Texas Instrument. (Jul. 2016). *MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide*. [Online]. Available: http://www.ti.com/lit/ug/slau367k/slau367k.pdf
[49] S.-Y. Wong and C. Chen, "Power efficient multi-stage CMOS rectifier design for UHF RFID tags," *Integr. VLSI J.*, vol. 44, no. 3, pp. 242–255, 2011.
[50] Sensor Systems Laboratory. (Jun. 2011). *WISP Code Examples*. [Online]. Available: http://wisp.wikispaces.com/WISP+code+examples
[51] J. R. Smith, *Wirelessly Powered Sensor Networks and Computational RFID*. New York, NY, USA: Springer, 2013.
[52] Microchip. (Jan. 2003). *24AA08/24LC08B, 8K I2C Serial EEPROM*. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/21710c.pdf

Die Wu (S'16) received the B.E. degree in information security from the University of Electronic Science and Technology of China in 2011. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Electronic Science and Technology of China. His research interests include RFID systems, wireless networks, and pervasive computing.



Li Lu (M'07) received the B.E. and M.S. degrees in automation control in 2000 and 2003, respectively, and the Ph.D. degree from the Key Lab of Information Security, Chinese Academy of Science, in 2007. He is currently an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include RFID technology and system, wireless network, and network security. He is a member of ACM.



Muhammad Jawad Hussain received the B.E. degree in avionics from the National University of Science and Technology, Pakistan, in 2005, and the M.S. degree in information security from the University of Electronic Science and Technology of China in 2013. He is currently pursuing the Ph.D. degree in communication and information engineering with the University of Electronic Science and Technology of China. His research interests include security in RF backscatter tokens, computational RFIDs, and structural health monitoring systems.

Wenyu Yang received the B.E. and M.S. degrees in computer engineering from the University of Electronic Science and Technology of China, in 2013 and 2016, respectively. Her research interests include RFID systems and network security.