Hardware Support for Secure Intermittent Architectures

Archanaa S. Krishnan
Virginia Tech

Patrick Schaumont
Virginia Tech

***Abstract*—Intermittent computing applications in the IoT space, such as long-term monitoring of the structural integrity of infrastructures, rely on battery-less computing systems powered through scavenged energy. For such systems, power-loss is a fact of life, and there is a need for a secure power transition mechanism to convert the active system state into a protected non-volatile form and back. We evaluate the architectural needs to secure these power transitions and to adapt computations based on the scavenged energy. Our objective is to enforce confidentiality, integrity, freshness, and authenticity over the system state across power loss. We observe that secure power transitions are delicate and complex. We need secure checkpoints which are expensive to compute, and which may require hardware-accelerated cryptography and isolated secure non-volatile storage. Next, we observe that in intermittent systems, the energy subsystem does not adapt to the needs of the application. Rather, the application must adjust its computing pattern to the available energy. We define an energy-harvester subsystem interface to optimize the run-time activity of the intermittent system. The interface drives the optimized execution of a secure communication protocol (covering key-exchange and bulk encryption), such that wasted energy is eliminated and that run-time performance is improved. We report results from several prototyping experiments.**

I. Intermittent Architectures

Traditional architectures are conceived from the viewpoint that power is plentiful and that power will only be fully removed when all tasks are completed. The power management is adjusted to the needs of the application or to the computing load. Yet in systems where the power source is unreliable and limited, the architecture adapts to the available power, including seamless and transparent turn-off and turn-on. Such architectures are intermittent - they can pause computation to save and later restore system state. Intermittent architectures use scavenged or harvested energy sources, which provide a nearly inexhaustible energy supply with limited and unreliable power delivery (think of a solar cell). Depending on the energy scavenging source, the power level can be as low as a few microwatt. Through power conditioning, scavenged energy is stored in an energy buffer, C_B, which in turn has limited capacity and which may overflow. This makes continuous operation of an architecture virtually impossible; at some point, the energy buffer runs out. However, by saving critical system state as a *checkpoint*, system operation can continue across power loss. The checkpoint generation is either triggered by a system call in volatile processors or automatically triggered by a power interrupt in non-volatile processors [1]. Non-volatile processors store a majority of their data in non-volatile memory and place their system data, such as registers, in volatile memory. They only have to back up the volatile state in non-volatile logic, thus have the advantage of instant state restore on power-up with negligible energy and time overhead.

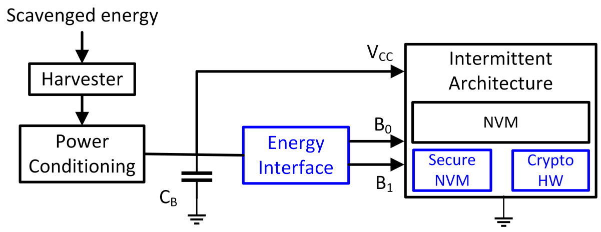

Fig. 1. Secure intermittent architecture with the expected hardware support for energy interface, secure non-volatile memory, NVM, and cryptographic hardware. C_B: Energy buffer; V_{CC}: Power supply; B_0, B_1: 2-bit interface to indicate the energy level indicator in C_B.

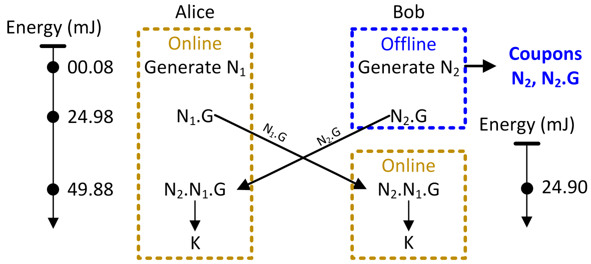

Fig. 2. DH key exchange between Alice and Bob. Alice requires 49.88mJ to compute the shared secret, $N_1.N_2.G$. Pre-computing the random number, N_1, and the point multiplication $N_1.G$ reduces Bob's online energy requirement to 24.9mJ.

II. Security and Energy

Since checkpoints may contain critical settings such as kernel privileges, memory access rights, and cryptographic keys, they must provide the following security guarantees.

- Confidentiality, integrity, and authenticity, to protect checkpoints against unauthorized reads and writes.
- Freshness, to protect checkpoints against replay.
- Atomicity, to ensure that the checkpointing itself is robust against power loss.

When the scavenged energy is in excess of the application needs, it cannot be accumulated beyond the capacity of C_B. The application is made aware of the excess energy, and preomputes coupons with it to avoid energy wastage. Coupons

TABLE I
OVERHEAD OF THE ONLINE PORTION OF REGULAR AND PRECOMPUTED CRYPTOGRAPHIC PRIMITIVES[1]

Crypto primitive	Coupon Size	Regular		Precomputed	
		Energy (uJ)	Time (ms)	Energy (uJ)	Time (ms)
AES-CTR	128 bytes	17.8	12.4	8.7	6.30
TRNG	256 bits	79.8	68.2	0.7	$\sim$[2]
DH Key exchange	256 bits	49880.0	43868.2	24900	21900

[1] Measurements are from MSP430FR5994 operating at 1MHz.
[2] Time taken to read a 256-bit value from FRAM (0.7μs)is negligible.

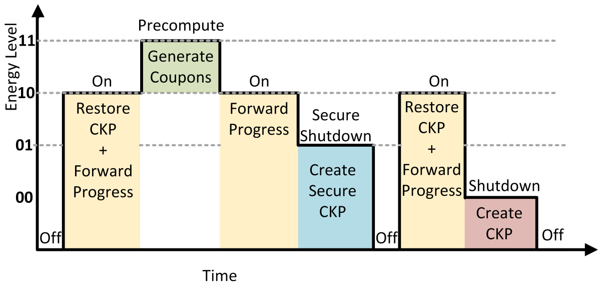

Fig. 3. Different levels of the energy interface and its corresponding functions in the architecture.

are the intermediate results of data independent portions of an algorithm. Cryptographic algorithms are ideal for precomputation as they pre-process static data, such as Diffie Hellman(DH) key exchange, illustrated in Figure 2. Alice and Bob agree upon a common elliptic curve with a base point, G, to establish a secret key, K, over an insecure channel in three steps. First, they each compute a random number; second, they multiply it with G; and third, they exchange the product and multiply the incoming product with their private random number to arrive at the shared secret, $N_1.N_2.G$, from which the secret key, K, is derived. The first two steps are independent of the input data; Bob precomputes the first two steps when excess energy is available offline, and stores N_2 and $N_2.G$ as coupons. At run-time, Bob extracts the coupons from memory, communicates the product and computes the shared secret with the input. Bob only requires 2409mJ of energy to compute the shared secret, whereas, Alice requires 49088mJ because she did not precompute the first two steps during the offline phase. Precomputation is also applicable to other cryptographic algorithms, such as digital signatures [2] and bulk encryption [3]. Since it generates cyptographic states as coupons, which are stored in non-volatile memory, the coupons must be protected along with the critical system data in secure checkpoints.

III. Hardware Support

A secure intermittent architecture requires hardware support to generate and restore secure checkpoints, and to adapt its computation based on the energy level in C_B, which are illustrated in Figure 1.

a) Secure Checkpoints: Secure non-volatile storage can be used to store checkpoints, which prevents unauthorized reads and writes to the stored data, both during power on and power off. The size of secure storage is fixed to an architecture, for example the Zatara ZA9L1 provides 4kB of secure storage [4]. Since the checkpoint size depends on both the architecture and application, it maybe larger than the available secure storage. With the secure storage as root of trust, a dedicated protocol is designed to protect the checkpoints in insecure non-volatile memory, which lacks any protection [5]. The protocol introduces a nonce to every checkpoint for freshness, which is placed in secure storage. The checkpoint is then encrypted, authenticated, and stored in insecure non-volatile memory. An unsecured checkpoint only requires 0.003μJ/bit to write to non-volatile memory but it leaves the power transition vulnerable. Whereas, a secure checkpoint provides all the security guarantees and costs 22.2μJ/bit, which includes the energy required for cryptographic operations and write to non-volatile memory. Accelerated cryptographic hardware modules reduce the latency and energy required to secure checkpoints.

b) Energy Interface: The intermittent system turns on when the input power supply reaches a preset V_{CC}. After power-on, the forward progress of the application adjusts itself based on the input from the energy interface. The interface, in Figure 1, indicates the level of energy in C_B using two bits, B_0 and B_1. It provides four levels of energy - 00, 01, 10, and 11, depicted in Figure 3, where each level corresponds to an adapted computing pattern.

We use this interface with an intermittent architecture to establish a secure communication link. First, a key is exchanged over an insecure channel using DH key exchange in Figure 2. Next, using this key, all future messages through this channel are encrypted using a bulk encryption algorithm, such as AES in counter mode. The random number, point multiplication, and key stream for bulk encryption are precomputed when the device is idle and the energy level is '11'. Secure checkpoint calls are triggered when the energy level is '01'. Table I lists the overhead of the online phase of regular and precomputed versions of the cryptographic primitives used in establishing the link.

References

[1] S. Khanna, S. C. Bartling, M. Clinton, S. Summerfelt, J. A. Rodriguez, and H. P. McAdams, "An FRAM-Based Nonvolatile Logic MCU SoC Exhibiting 100% Digital State Retention at $V_{DD} = 0$ V Achieving Zero Leakage With 400-ns Wakeup Time for ULP Applications," *IEEE J. of Solid-State Circuits*, vol. 49, no. 1, pp. 95–106, Jan 2014.

[2] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signatures," *J. Cryptology*, vol. 9, no. 1, pp. 35–67, 1996.

[3] W. Shi, H. S. Lee, M. Ghosh, C. Lu, and A. Boldyreva, "High efficiency counter mode security architecture via prediction and precomputation," in *32st Inter. Symp. on Computer Architecture (ISCA 2005), 4-8 June 2005, Madison, Wisconsin, USA*, 2005, pp. 14–24.

[4] "Zatara High-Performance, Secure,32-Bit ARM Microcontroller," Maxim, Tech. Rep., March 2009, available at https://datasheets.maximintegrated.com/en/ds/ZA9L1.pdf.

[5] A. S. Krishnan, C. Suslowicz, D. Dinu, and P. Schaumont, "Secure intermittent computing protocol: Protecting state across power loss," in *Design Automation and Test in Europe(DATE 2019), Florence, Italy, March 2019.*