A Low-Power BLS12-381 Pairing Crypto-Processor for Internet-of-Things Security Applications

Utsav Banerjee, Member, IEEE, and Anantha P. Chandrakasan, Fellow, IEEE

Abstract—We present the first BLS12-381 elliptic curve pairing crypto-processor for Internet-of-Things (IoT) security applications. Efficient finite field arithmetic and algorithm-architecture co-optimizations together enable two orders of magnitude energy savings. We implement several countermeasures against timing and power side-channel attacks. Our crypto-processor is programmable to provide the flexibility to accelerate various elliptic curve and pairing-based protocols such as signature aggregation and functional encryption.

Index Terms—Elliptic Curve Cryptography (ECC), Pairing-Based Cryptography (PBC), cryptographic accelerator, hardware security, low-power, side-channel, Internet of Things (IoT).

I. INTRODUCTION

Elliptic curves are used as the de facto standard for traditional public key cryptography such as key establishment, digital signatures, authenticated key exchange and public key encryption [1]. Pairing-based cryptography (PBC), a variant of elliptic curve cryptography (ECC), uses bilinear maps between elliptic curves and finite fields to enable novel applications beyond traditional key exchange and signatures [2]. Fig. 1 shows two such applications – (a) signature aggregation, where arbitrarily large number of signatures are compressed into one to resolve communication bottleneck in mesh networks such as blockchain [3], and (b) functional encryption, which allows computing on encrypted data with a function embedded in the decryption key. In particular, pairing-based function-hiding inner product encryption [4] allows computing the inner product of two encrypted vectors. This can be used for simple privacy-preserving data classification tasks, thus enabling a new paradigm in the field of secure computation.

Only special pairing-friendly elliptic curves can be used for pairing-based cryptographic protocols. The security of commonly used BN-254 and BN-256 pairing-friendly curves (based on 254b and 256b prime fields respectively) has been compromised by recent advances in cryptanalysis [5]. The BLS12-381 pairing-friendly elliptic curve, based on a 381b prime field, has been recently proposed and is part of ongoing standardization led by the Internet Engineering Task Force (IETF) [5]. Along with strong security, the new curve also has higher computational complexity, thus making it challenging to implement on low-power embedded devices. While previous work [6]–[8] implement hardware for 254b and 256b BN curves, efficient hardware accelerators for BLS12-381 are largely unexplored. In this work, we present a low-power BLS12-381 elliptic curve pairing crypto-processor, and this is the first ASIC implementation supporting the BLS12-381 curve (extended version of [9]). Our design enables two orders of magnitude energy savings through efficient hardware acceleration, implements countermeasures against timing and power side-channel attacks, and provides flexibility to implement various ECC and PBC protocols for IoT applications.

II. BLS12-381 HARDWARE IMPLEMENTATION

The pairing computation is a bilinear map $e: G_1 \times G_2 \rightarrow G_T$, where $G_1$, $G_2$ are elliptic curve groups and $G_T$ is a finite field group. This map satisfies the bilinear property $e(aP, bQ) = e(P, Q)^{ab}$, where $P \in G_1$, $Q \in G_2$, $a, b \in \mathbb{Z}_q^*$, $q$ being the group order (a prime). This is the bilinear property which enables novel pairing-based cryptographic protocols.

For BLS12-381, the groups $G_1$ and $G_2$ are based on elliptic curves $E(\mathbb{F}_p^*) : y^2 = x^3 + 4$ and $E'(\mathbb{F}_{p^2}) : y^2 = x^3 + 4(1+\alpha)$ respectively and $G_T$ is based on the extension field $\mathbb{F}_{p^4}$, where $p$ is a 381-bit prime and the order $q$ of each group is a 255-bit prime [5].

A. Finite Field Arithmetic

Software profiling indicates that big integer prime field arithmetic, especially modular multiplication, accounts for more than 90% of PBC computation cost. For BLS12-381, we need arithmetic over the 381-bit prime field $\mathbb{F}_p$ (base field) and the 255-bit prime field $\mathbb{F}_q$ (scalar field).

Our modular adder-subtractor design, shown in Fig. 2, consists of a pair of cascaded 381b adder-subtractors. The
Unlike traditional ECC, pairing computation also requires arithmetic over extensions of the field $\mathbb{F}_p$. For BLS12-381, these are $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)/(\alpha^2 + 1)$, $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}(\beta)/(\beta^3 - 1 - \alpha)$ and $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}(\gamma)/(\gamma^2 - \beta)$ [5]. This construction of the form $\mathbb{F}_p \rightarrow \mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^6} \rightarrow \mathbb{F}_{p^{12}}$ is known as towered arithmetic. Extension field arithmetic over $\mathbb{F}_{p^2}/\mathbb{F}_p$, $\mathbb{F}_{p^6}/\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^{12}}/\mathbb{F}_{p^6}$ involves manipulation of polynomials with coefficients in $\mathbb{F}_p$.

We speed up multiplications, squarings and inversions in these extension fields by extensively using Karatsuba-style divide-and-conquer techniques [2], which provides 35% reduction in pairing energy consumption [11].

### B. Elliptic Curve and Pairing Computations

We use homogeneous projective coordinates for all elliptic curve point operations. To prevent side-channel vulnerabilities, we employ optimized exception-free point doubling and complete point addition formulas based on [12]. This ensures that the implementation is constant-time and avoids any data-dependent conditional executions.

Elliptic curve scalar multiplication (ECSM) is implemented using the double-and-add-always algorithm to prevent timing side-channel leakage [13]. Our implementation of constant-time ECSM with a 255b scalar requires $4,847M_1 + 14,025A_1 + I_1$ (where $I_1 = 608M_1$) for $G_1$ and $4,337M_2 + 510S_2 + 10,200A_2 + I_2$ (where $I_2 = 4M_1 + 2A_1 + I_1$) for $G_2$, where $A_1$ (resp. $A_2$), $M_1$ (resp. $M_2$) and $I_1$ (resp. $I_2$) denote additions / subtractions, multiplications / squarings and inversions respectively in $\mathbb{F}_p$ (resp. $\mathbb{F}_{p^2}$). ECSM performance can be improved by using standard pre-computation techniques (memory-time trade-offs) such as windowing, comb, etc [1].

The two main components of pairing computation are Miller Loop (ML) and Final Exponentiation (FE) [2]. The Miller Loop consists of a series of line computations based on binary representation of the curve parameter $u$. For our implementation of BLS12-381, their computation costs, in terms of equivalent number of $\mathbb{F}_p$ multiplications, are $7,050M_1$ and $8,339M_1$ respectively. The overall pairing involves 15,389 $\mathbb{F}_p$ multiplications and requires approximately 3.4M cycles.

Many practical pairing-based protocols require evaluating the product of several pairings, also known as multi-pairing:

$$\prod_{j=1}^{n} e(P_j, Q_j) = e(P_1, Q_1) \times e(P_2, Q_2) \times \cdots \times e(P_n, Q_n)$$

We share ML and FE computations across multiple pairing instances to speed up multi-pairing, which is especially useful in aggregate signature verification [3]. Sharing only the FE operation provides $2\times$ improvement, while sharing both ML and FE provides another 30% energy savings [11].

Function-hiding inner product encryption [4] requires scalar multiplications on $\mathbb{G}_2$. The corresponding twist curve $E'$ supports the skew Frobenius map [14], which allows efficient computation of $\hat{\phi}(P) = pP \forall P \in E'(\mathbb{F}_{p^2})$. To speed up the encryption step, we split the 255-bit scalar $k < q$ into two 128-bit parts as $k = k_1 + k_2u^2$ where $p = u (mod \ q)$ and compute $kP = k_1P + k_2(u^2P) = k_1P + k_2\hat{\phi}(\hat{\phi}(P))$ using multi-exponentiation [1]. This leads to $1.8\times$ energy savings in inner product encryption per vector element [11].
Pairing-based signatures also require mapping random $\mathbb{F}_p$ elements to points on elliptic curves, known as the hash-to-curve operation [3]. Our implementation of hash-to-$G_1$ requires 1,897 $\mathbb{F}_p$ multiplications. Since our modular arithmetic unit can be configured for the 255-bit prime $q$, we also support ECC using Jubjub, a twisted Edwards curve over $\mathbb{F}_q$ [5]. Our constant-time Jubjub ECSM requires 4,755 $\mathbb{F}_q$ multiplications.

### III. PAIRING CRYPTO-PROCESSOR ARCHITECTURE

The top-level architecture of our pairing crypto-processor is shown in Fig. 4, where the arithmetic unit is integrated with a 15.375 KB data memory, a 1 KB instruction memory and an instruction decoder. It can be programmed using 32b custom instructions to perform different modular arithmetic, elliptic curve and pairing computations.

The crypto-processor data memory is hierarchical with three levels. First, the modular arithmetic unit is coupled with a small $8 \times 384$-bit register file $M_0$ implemented using flip-flops for efficiency. $M_0$ is used for all $\mathbb{F}_p$ and $\mathbb{F}_p^2$ arithmetic computations. At the next level, a $64 \times 384$-bit SRAM $M_1$ is used to store all temporary variables required for $\mathbb{F}_p^6$, $\mathbb{F}_p^{12}$, $G_1$, $G_2$ and $G_T$ arithmetic computations. Finally, a $256 \times 384$-bit SRAM $M_2$ is used to store protocol-level inputs and outputs. While simple ECSM and pairing computations require only a few of these 256 memory locations in $M_2$, having a large top-level memory is useful to support efficient multi-pairings and hash-to-curve maps. Each memory module is dynamically clock gated based on the function under execution, providing up to 20% power savings. Fig. 5 shows simulated waveforms for memory clock gating during final exponentiation computation. The modular arithmetic unit is operational continuously, while the memories are accessed only when data movement is required.

The pairing computation requires several constants in Montgomery domain, which are stored in a $22 \times 384$-bit lookup table (LUT). The $\mathbb{F}_p$ and $\mathbb{F}_p^2$ functions are handled by the modular arithmetic unit. Optimized micro-code for $\mathbb{F}_p^6$, $\mathbb{F}_p^{12}$, $G_1$, $G_2$ and $G_T$ arithmetic functions (which require $\mathbb{F}_p$ and $\mathbb{F}_p^2$ arithmetic) are stored in another $768 \times 24$-bit LUT. To save area, these LUTs are realized entirely using digital logic. Combined area of these two LUTs is only 6k-gate, which is 53k-gate and 34k-gate smaller than SRAM-based and ROM-based implementations respectively.

### IV. SYSTEM DESIGN AND MEASUREMENT RESULTS

#### A. Chip Architecture and Test Setup

As shown in Fig. 6, the pairing crypto-processor is integrated (through a memory-mapped interface) with a low-power RISC-V micro-processor supporting the RV32IM instruction set, with 1-cycle multiplier, 32-cycle divider, 32 KB instruction memory and 64 KB data memory [15]. The RISC-V is also coupled with accelerators for AES-128/256 and SHA2-256 [16]. Reading from and writing to accelerators through the memory-mapped interface require 3 cycles and 2 cycles respectively. The RISC-V, AES, SHA and pairing cores all have dedicated clock gates independently configurable for power savings. The RISC-V core can be clock-gated using wait-for-interrupt instruction, and it is woken up by dedicated interrupts from the cryptographic accelerators. The accelerators can be accessed through software using simple load and store instructions, without any changes to the compilation toolchain. The RISC-V processor is used for scheduling cryptographic workloads as well as for processing their inputs and outputs.

Our test chip (Fig. 7) was fabricated in the TSMC 40nm low-power process. Our pairing crypto-processor consists of 112k logic gates and 16 KB SRAM, with a total area of 0.2 mm$^2$ (logic and memory combined). The RISC-V (including
interrupt controller and peripherals), AES and SHA cores occupy 52k, 12k and 23k logic gates respectively. Our test chip supports supply voltage scaling from 0.66 V to 1.1 V, and its maximum operating frequency (for both RISC-V and accelerators) at 0.66 V and 1.1 V are 16 MHz and 90 MHz respectively. Fig. 7 shows our measurement setup. An FPGA is used to transfer instructions / programs from a host computer to the instruction memory of our test chip. All elliptic curve and pairing-based cryptography programs are written using custom instructions and compiled with a Python script, which is then integrated together with the RISC-V software (Fig. 8).

Fig. 9 compares our design with previous work on pairing hardware accelerators. AES and SHA cores occupy 52k, 12k and 23k logic gates respectively. Our test chip supports supply voltage scaling from 0.66 V to 1.1 V, and its maximum operating frequency (for both RISC-V and accelerators) at 0.66 V and 1.1 V are 16 MHz and 90 MHz respectively. Fig. 7 shows our measurement setup. An FPGA is used to transfer instructions / programs from a host computer to the instruction memory of our test chip.

B. Protocol Implementation Results

To measure the efficiency of our design as well as to demonstrate its flexibility in supporting various security applications, we have implemented and profiled several BLS12-381 pairing-based cryptography protocols on our test chip, as detailed in Fig. 10. Our hardware-accelerated implementations are 130-140× more energy-efficient compared to software. All measurement results are reported at 16 MHz and 0.66 V, the operating condition providing lowest energy consumption. Effect of voltage-frequency scaling is shown in Fig. 11.

C. Side-Channel Countermeasures

As countermeasures against timing and simple power analysis (SPA) side-channel attacks [13], we use complete point addition formulas [12] and double-and-add-always technique [17] in our ECSM and pairing implementations. All results discussed earlier are from constant-time implementations with these countermeasures. To prevent stronger differential power analysis (DPA) attacks, we employ the following countermeasures:

- Randomized projective coordinates [18], [19], where elliptic curve points \((X : Y : Z)\) are transformed into the form \((\lambda X : \lambda Y : \lambda Z)\) with non-zero random \(\lambda \in \mathbb{F}_p\).
- ECSM with random scalar splitting [17], where secret scalar \(k \in \mathbb{F}_q\) is split into two parts \(r\) and \(k - r\) with random \(r \in \mathbb{F}_q\), computed as \(kP = rP + (k - r)P\).
- Pairing with random exponents and bilinear property [19], computed as \(e(aP, bQ) = e(P, Q)^{ab} = e(P, Q)\) with random \(a \in \mathbb{F}_p\) and \(b = a^{-1} \mod q\).

The first technique is practically free, requiring only a few \(\mathbb{F}_p\) multiplications. The second technique can be significantly simplified by using multi-exponentiation [1], where \(2P\) is pre-computed and both scalars \(r\) and \(k - r\) are processed simultaneously to share point doublings and merge point additions. The third technique is quite expensive, requiring one \(\mathbb{G}_1\) ECSM, one \(\mathbb{G}_2\) ECSM and one \(\mathbb{F}_q\) inversion.
also require generation of random elements in $\mathbb{F}_p$ or $\mathbb{F}_q$, performed by using our SHA2-256 accelerator as a cryptographically secure pseudo-random number generator. Fig. 12 shows measured power side-channel leakage assessment results (based on standard non-specific fixed vs. random $t$-test [20]) over 0.5M traces for our hardware implementation with all countermeasures. Here, $|t| < 4.5$ indicates, with very high confidence, that there is no evidence of first-order side-channel leakage. We have also performed difference-of-means test to verify that there is no significant observable information leakage through power consumption for any bit in ECJM secret scalar being 0 versus 1. For ECJM, the DPA countermeasures have only 10% performance and energy overheads. However, DPA countermeasures lead to $2.3 \times$ additional overhead for pairing. We note that all these countermeasures can be implemented without any changes to our hardware, by utilizing the programmability of our crypto-processor.

V. CONCLUSION

In this work, we have presented a low-power programmable crypto-processor to accelerate ECC and PBC using the recently proposed BLS12-381 pairing-friendly elliptic curve. Using optimized algorithms and low-power area-efficient architectures, we demonstrate practical hardware-accelerated pairings which enable novel cryptographic protocols to secure resource-constrained IoT devices.

ACKNOWLEDGMENT

The authors would like to thank Texas Instruments for funding this work, the TSMC University Shuttle Program for chip fabrication support, and BlueSpec, Xilinx, Cadence, Synopsys and Mentor Graphics for providing CAD tool support.