ECG-based Authentication using Timing-Aware Domain-Specific Architecture

Renato Cordeiro, *Member, IEEE,* Dhruv Gajaria, *Graduate Student Member, IEEE,* Ankur Limaye, *Graduate Student Member, IEEE,* Tosiron Adegbija, *Senior Member, IEEE,* Nima Karimian, *Member, IEEE,* and Fatemeh Tehranipoor, *Member, IEEE*

***Abstract*—Electrocardiogram (ECG) biometric authentication (EBA) is a promising approach for human identification, particularly in consumer devices, due to the individualized, ubiquitous, and easily identifiable nature of ECG signals. Thus, computing architectures for EBA must be accurate, fast, energy-efficient, and secure. In this paper, first, we implement an EBA algorithm to achieve 100% accuracy in user authentication. Thereafter, we extensively analyze the algorithm to show the distinct variance in execution requirements and reveal the latency bottleneck across the algorithm's different steps. Based on our analysis, we propose a domain-specific architecture (DSA) to satisfy the execution requirements of the algorithm's different steps and minimize the latency bottleneck. We explore different variations of the domain-specific architecture, including one that features the added benefit of ensuring constant timing across the different EBA steps, in order to mitigate the vulnerability to timing-based side-channel attacks. Our DSA improves the latency compared to a base ARM-based processor by up to 4.24x, while the constant timing DSA improves the latency by up to 19%. Also, our DSA improves the energy by up to 5.59x, as compared to the base processor.**

***Index Terms*—Domain-specific architectures, energy efficient, secure architectures, side-channel attacks, ECG, biometric authentication, Internet of Biometric Things (IoBT), Internet of Things (IoT)**

I. Introduction and Motivation

Consumer devices, such as smartphones and wearables, have become the fastest-growing category of Internet of Things (IoT) devices. Many of these devices also constitute the Internet of Medical/Health Things (IoMT/IoHT), which enable innovative healthcare solutions and services. With the rapid growth of the IoT, fast, energy-efficient, and secure user authentication has become a necessity for consumer IoT devices. IoMT devices, especially, are prone to cyber-attacks and adversarial threats due to their interaction with sensitive and private user information. As such, there is much ongoing research into modalities for accurate and efficient user authentication in consumer devices [1], [2].

R. Cordeiro and N. Karimian are with the Department of Computer Engineering, San Jose State University, USA (email: {renato.silveiracordeiro,nima.karimian}@sjsu.edu)

D. Gajaria, A. Limaye, and T. Adegbija are with the Department of Electrical and Computer Engineering, University of Arizona, USA (email: {dhruvgajaria,ankurlimaye,tosiron}@email.arizona.edu)

F. Tehranipoor is with the Department of Electrical and Computer Engineering, Santa Clara University (email: ftehranipoor@scu.edu)

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis 2020 and appears as part of the ESWEEK-TCAD special issue.

This work is supported in part by the National Science Foundation under grant CNS-1844952.

Electrocardiogram (ECG) is an increasingly popular modality for biometric user authentication. Apart from its value for deriving health data insights and diagnosis through healthcare monitoring [3], ECG enables user authentication based on physiological signals. ECG signals, which represent the human heart's electrical activity, are easy to obtain, uniquely identifiable, permanent, and information-rich, making them an excellent choice for user authentication in resource-constrained systems [1]. As a result, there have been prior studies on fast and energy-efficient EBA techniques, from the perspectives of both the algorithm and hardware implementation [2], [3].

Most prior hardware approaches to implementing EBA at extremely low time and energy typically involve application-specific integrated circuits (ASIC) or FPGA-based designs [4], [5]. While these approaches achieve high speed and energy efficiency, they are inflexible and can only function for the specific EBA algorithm that the hardware is designed for. A modification of the algorithm would require a re-implementation of the hardware, potentially from scratch [3]. Since EBA is not a persistent process, dedicated hardware just for biometric authentication may be redundant, especially in resource-constrained general- (or multi-) purpose devices, such as smartphones or smartwatches. Furthermore, there is currently a dearth of approaches that consider the security of EBA architectures in consonance with the need for energy-efficiency and low latency. Given the high variability in the timing and power profile of different phases of the ECG authentication [6] algorithm, these implementations are prone to timing or power side-channel attacks [7], [8], [9].

In this work, we propose a *domain-specific architecture (DSA)* for ECG biometric authentication. As an important step in the direction of efficient and secure architectures for EBA, our work aims to mitigate *timing-based side-channel attacks* by ensuring that the different steps of the authentication process exhibit both *intra-step* and *inter-step* constant timing profiles. As such, the constant timing mitigates the variability and timing leakage required for performing timing-based attacks. Our main motivation for designing a DSA rather than an ASIC or FPGA-based design is the runtime flexibility afforded by domain-specific architectures. When the EBA algorithm is not running, other processes or threads can be run on the system and can be preempted for the EBA algorithm when necessary.

We explore different versions of our DSA to evaluate their latency and energy benefits and their tradeoffs. We explore a design featuring a dedicated buffer to mitigate the latency and energy overheads of data movement and one featuring a custom block to mitigate the performance bottleneck in the EBA algorithm. We also explore an adaptable DSA (via dynamic frequency scaling [10] and adaptable execution order [11]) to specialize the execution resources to the variable runtime needs of the EBA algorithm while also achieving constant timing.

Our main contributions are summarized as follows:

- We analyze the execution characteristics of the EBA algorithm to reveal the performance bottleneck and variable timing characteristics that increase their vulnerability to timing-based side-channel attacks.
- We design an energy-efficient domain-specific architecture (DSA) to reduce the execution time of EBA. To mitigate the vulnerability to timing-based side-channel attacks, our architecture can also maintain constant timing across the different authentication steps by trading off optimization potential.
- We explore different design variants of the DSA and analyze their tradeoffs. We compare the DSA to a baseline ARM processor configuration commonly found in modern smartphones and show that the proposed architectures offer substantial latency and energy improvements over the baseline architecture. The adaptable DSA can reduce the latency and energy by up to 19% and 4.62x, respectively, while maintaining constant timing across the algorithm's steps. Furthermore, our DSA eliminates the latency overheads imposed by prior work that also used constant timing to mitigate vulnerability to timing-based side-channel attacks.

The rest of the paper is organized as follows. In Section II, we describe ECG biometric authentication, various steps of EBA, and details of our implementation of the algorithm. In Section III, we discuss domain-specific architectures for EBA, details of our DSA design, timing-based side-channel attacks in EBA, the threat model considered in this paper, and mitigation methods. We describe our experimental setup and experimental results in Section IV and Section V, respectively. Section VI provides a brief literature review of related work. Finally, in Section VII, we present our conclusion and overview of future work.

II. ECG Biometric Authentication (EBA)

Electrocardiogram signals result from the human heart's electrical activities. Many authentication schemes in the healthcare domain, including the burgeoning IoMT, utilize ECG biometrics for authentication [12], due to several advantages, such as internal security, ease of implementation, liveness detection, etc.

EBA systems, in general, apart from the sensor, are comprised of four major steps: *filtering, segmentation, feature extraction,* and *matching*. The ECG sensor provides the interface between the user and the authentication system, and collects the user's biometric traits. *Filtering* processes the gathered ECG signal to remove various noise sources in order to enhance the quality of the biometric traits. *Segmentation* splits the ECG signal into its different unique component

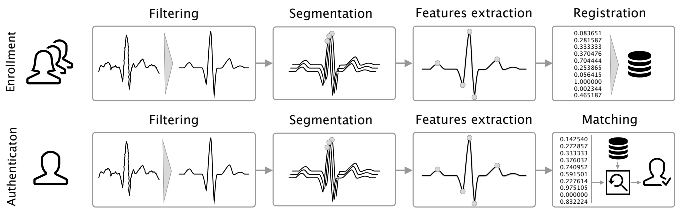


Fig. 1: High-level overview of an ECG biometric authentication (EBA) system.

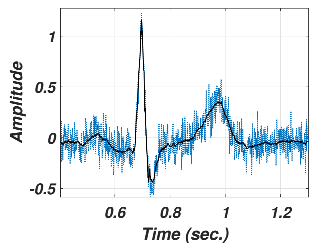
waveforms in order to reduce redundancy and simplify the authentication process. *Feature extraction* extracts information that may enable the system to distinguish between different users. The feature set extracted during an a priori enrollment phase is either stored in a remote database as a template indexed by the user's identity information (i.e., match-on server) or stored on a smart card (i.e., match-on card/device). This template will be computed by averaging a set of n enrolled ECG signals of the same user in the feature sets. *Matching,* which can be implemented in hardware or software, compares the template with a new input query and provides a response to the query, i.e., whether the user's biometric matches the template or not.

A. EBA Algorithm

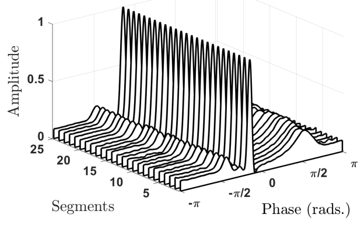
Fig. 1 provides a high-level overview and flow of the EBA process, which comprises of the *enrollment* and *authentication* phases. In the enrollment phase, the user's ECG signal is registered to generate the template, and in the authentication phase, which this work focuses on, raw data from a user is provided and compared to the previously stored template to determine the access permissions. In what follows, we briefly describe the various steps of the EBA algorithm and our approach for implementing the algorithm.

Data Acquisition: ECG measures electrical activity in the heart and electrical signals produced during muscle contractions. Generally, an ECG sensor consists of two electrodes. In early research work, wet electrodes such as AD8232 were dominant, while the recent advancements in sensing technologies make the use of dry electrodes more feasible than ever. The Nymi wristband [13] and CardioWheel [14] are examples of commercial ECG sensing products that have been developed to improve the wearer's daily experience.

Filtering: The presence of noise within the signals might result in inaccurate results. Hence, denoising is a required step in EBA systems. Different types of noise get assembled together with ECG signals in the process of acquisition and transmission. These signals can range from low-frequency noise such as baseline wander (BW) to high-frequency noises such as power line transmission motion artifact (MA) and electrode movement (EM). To remove this artifact, we employed an infinite impulse response (IIR) bandpass filter by cascading a low-pass (LP) and high-pass (HP) filters with cutoff frequency 1Hz-40Hz, as shown in Fig. 2a. We aimed to preserve the useful original information of the ECG while attenuating low and high-frequency noise components. The



(a)



(b)

Fig. 2: Plots illustrating ECG signal from the database for (a) filtered vs. noisy ECG. (b) ECG segments collected from the same subject and localization of fiducial points. After detecting R-peak using the Pan-Tompkins algorithm, the ECG signal is segmented by cardiac cycles.

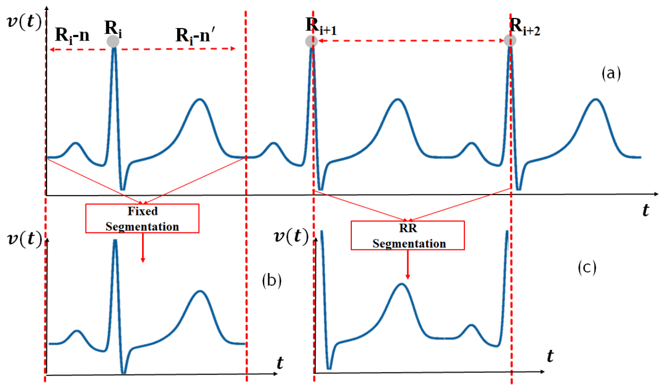


Fig. 3: Hypothetical example representing the ECG segmentation. (a) is the original ECG waveform, (b) illustrates fixed length segmentation, and (c) illustrates the RR segmentation techniques

.

high-pass filter with the transfer function of the second-order low-pass filter is:

$$H(z) = \frac{(1-z^{-6})^2}{(1-z^{-1})^2} \tag{1}$$

The amplitude response is:

$$|H(wT)| = \frac{\sin^2(3\omega T)}{\sin^2(\omega T/2)} \tag{2}$$

where T is the sampling period. The difference equation of the filter is:

$$\begin{aligned} y(nT) =& 2y(nT-T) - y(nT-2T) + x(nT) \\ &- 2x(nT-6T) + x(nT-12T) \end{aligned} \tag{3}$$

In addition, the transfer function for a high-pass filter is defined by the following equation:

$$H(z) = \frac{(-1+32z^{-16}+z^{-32})}{(1+z^{-1})} \tag{4}$$

where the difference equation is defined as follows:

$$\begin{aligned} y(nT) =& 32x(nT-16T) - [y(nT-T) \\ &+ x(nT) - x(nT-32T)] \end{aligned} \tag{5}$$

Segmentation: A typical ECG tracing consists of a series of P, QRS, and T waveforms occurring in a repetitive order. Each cycle of ECG can provide the same information over time, and it is not efficient to repetitively read correlating signals. Therefore, segmentation has come to the forefront of ECG biometric systems. The goal of segmentation is to find repeated patterns in the P, QRS, and T waveforms, thereby significantly reducing the template size in order to simplify template matching. The first step of ECG segmentation is to identify the R-peak. To achieve this goal, we employed the Pan-Tompkins [15] technique to detect ECG R-peak. In short, four steps including derivation, squaring, averaging phases before thresholds are set to identify R-peak for the ECG segmentation. We used a derivative filter to find the high slopes and identify the direction of the slopes of the ECG signal. The derivative filter also distinguishes the R-peak from other ECG waveforms. Squaring makes all the ECG signal values positive and amplifies the output of the previous stage. Averaging phase maximizes the ECG signal compared with the squared output. After the averaging process, the threshold is employed to detect the R-peaks in the ECG.

Upon successfully completing R-peak detection, the ECG signals are isolated into ECG beats (segments). There are two techniques for ECG segmentation called *Fixed Length Segmentation* and *RR segmentation*. Fixed length segmentation involves cropping the partial ECG signal at fixed distances before and after detected R-peaks ($R_{i-1} - n$, $R_{i-1} + n'$) instead of the whole signal, where n and n' are the time periods before and after the R-peak. *Note that n and n' are different from each other and vary depending on the sample rate of the data sets*. Finding an optimal value of n and n' plays a huge role in the EBA performance. Alternatively, RR segmentation involves cropping the whole waveform of the ECG signal (R_i, R_{i+1}), where the R_i is the ECG R-peak at cycle t, and R_{i+1} is the ECG R-peak at cycle $t+\tau$. In this work, we used *Fixed Length Segmentation* since we found it to give better results, allowing us to not only reduce the time for enrollment/authentication phase, but also reduce the memory space for storing the template. Fig. 3 illustrates our technique for segmenting ECG signals using sliding windows into different heartbeats.

Feature extraction: The feature extraction stage translates the segmented ECG into a representation that further reduces the effects of intra-subject variability while emphasizing discriminative and intra-class variations to obtain better performance. ECG biometric systems fall into two categories in terms of feature extraction: *fiducial point* or *non-fiducial point*. Fiducial point techniques focus on measurements of ECG fiducial landmarks in the time domain, such as temporal or amplitude difference between fiducial landmarks (P, QRS, and T). In non-fiducial techniques, on the other hand, feature extraction is based on using frequency analysis such as wavelet transform to holistically analyze an ECG signal and overall morphology of the waveform rather than specific fiducial points. Even though non-fiducial features such as frequency domain can lead to high accuracy, selecting the optimal level of decomposition and types of wavelet transform is challenging. Furthermore, if ECG signals are very noisy, non-fiducial techniques can degrade the EBA performance. Moreover, non-fiducial feature extraction is not lightweight and consumes more power, which is not ideal for IoMT [16]. Thus, in this paper, we use the fiducial method. Specifically, we extracted a subset of ten features that represent the majority of fiducial features from every beat of each individual's ECG signal. Fiducial point feature extraction relies on accurate detection of ECG fiducial characteristic points such as P, Q, R, S, and T waves, as shown in Fig. 4, in order to obtain their relative amplitude, temporal intervals, and morphological features [16]. Each temporal and amplitude of each waveform are distinctive from each individual user. In this work, we designed 10 fiducial points and 14 temporal features to generate a discriminative EBA feature representation to improve the EBA accuracy. To extract these features, first the R-peak, and then the P, Q, S, T peaks and valleys are detected using a local maximum/minimum searching algorithm within a defined physical region. *Note that the number of fiducial points and temporal features can be extended up to ≈ 40. However, we found that the aforementioned 24 feature sets were more robust against noises, discriminative, and achieved high accuracy.*

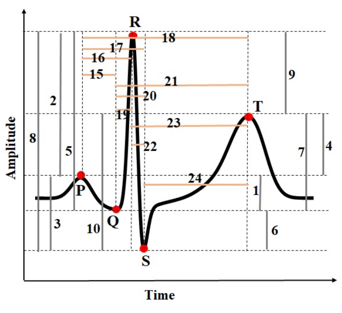
Matching: In the matching stage, identification and authentication functions are performed. Identification commonly includes a classification process such as support vector machines (SVM). For authentication, the acceptance or rejection of the identity claim is generally based on a reference threshold of T between the currently acquired trait and the previously acquired templates. In our work, we focus on authentication and employed Euclidean distance as a matching technique between the features' vectors to decide whether to accept or reject the identity claim. Given a claimed identity I and a query feature set X^q, we need to determine if (I, X^q) belongs to a genuine or imposter user. The Euclidean distance D between two feature vectors T_j and q_j is defined as

$$D(\{\mathbf{X_I^T}\}, \{\mathbf{X^q}\}) = \sqrt{\sum_{j=1}^{K} (\mathbf{X_I^T}[j] - \mathbf{X^q}[j])^2} \tag{6}$$

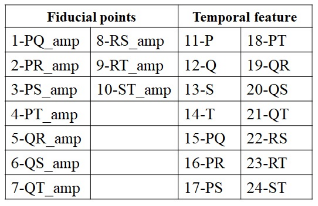
where X_I^T is a stored template corresponding to identity I and K is the number of feature sets. So, we compare X_I^T and X^q to measure the similarity for verification. If the distance D or score is above a predefined user-specific threshold (η), the claimed identity is accepted as a genuine user, otherwise, it is rejected and considered an imposter.

B. EBA Algorithm Evaluation

Evaluation metrics: To evaluate the performance of our EBA algorithm, we conducted the experiments with three error rates: false positive/accept rate (*FPR*/*FAR*), true positive/accept rate (*TPR*), and equal error rate (*EER*). *FRR* is the percentage of genuine users who were denied access to the ECG authentication system, whereas *FAR* is the percentage of imposters who successfully gained access to ECG biometric authentication. Both *FRR* and *TPR* can be traded-off with each other in order to find the optimal and desired *EER*. *EER* is the location on the receiver operator characteristic (ROC) curve where the *FAR* and true positive rate *1-FRR* are equal. Adjusting the threshold value (η) controls the *TPR* and *FPR* of the ROC curve. In fact, we generated a set of thresholds $\{\eta_j\}_{j=1}^{T}$ such



(a)



Fiducial points		Temporal feature	
1-PQ_amp	8-RS_amp	11-P	18-PT
2-PR_amp	9-RT_amp	12-Q	19-QR
3-PS_amp	10-ST_amp	13-S	20-QS
4-PT_amp		14-T	21-QT
5-QR_amp		15-PQ	22-RS
6-QS_amp		16-PR	23-RT
7-QT_amp		17-PS	24-ST

(b)

Fig. 4: (a) Single ECG beat with fiducial characteristic points, (b) temporal and fiducial point features that have been extracted from single ECG beat and used in this paper.

that $s_{\min} \leq \eta_j \leq s_{\max}, \forall j = 1, 2, \cdots, T$, where $s_{\min}$ and $s_{\max}$ are the maximum and minimum scores, respectively, in the given set of match score. Thus, each threshold (η_j) computes different values for *FAR* and *FRR* [17]. As (η_j) is decreased, the constraints on accuracy become more relaxed, allowing for higher *FAR*. For a relatively high threshold (η_j) value, the *FAR* is decreased. Depending on the application, the user-specific threshold (η_j) value can be adjusted.

We also calculated the accuracy for each subject as the number of successful attempts (segments or ECG beats) by the genuine user divided by the total number of attempts or the percentage of correctly recognized query samples. The accuracy is defined as follows:

$$Accuracy = \frac{N_c}{N_q} \tag{7}$$

where N_q is the total number of query samples and N_c is the number of query samples that are correctly identified.

ECG database: To validate the effectiveness of the EBA system, we conducted extensive experiments on three widely used benchmark datasets: ECG-ID [18], Combined measurement of ECG, Breathing and Seismocardiograms database (CEBSDB) [19], which contains normal ECG records, and PTB Diagnostic ECG Database [20], which contains both normal and abnormal ECG signals. The databases used are summarized in Table I. The PTB database contains 549 records with diverse profile information such as gender, age, healthy, unhealthy, and different lengths obtained from 290 subjects sampled at 1 kHz, which mimics real-world scenarios. Among the 290 subjects, 148 subjects showed serious abnormality, whereas there were 52 healthy subjects. All channels were involved, where only 14 are for ECG. However, in this work, we only used lead 1 as our experimental setup, and none of the people were excluded. ECG-ID contains twenty-second ECG recordings collected from 90 subjects from multiple sessions over a six-month period. The signals are acquired from single limb lead I using electrodes at the wrists. In this paper, we used all 90 subjects and used all sessions recordings ECG to conduct the experiments. In the CEBSDB database, 20 presumed healthy volunteers are measured using a Biopac MP36 data acquisition system from Santa Barbara, CA, USA. Total recordings are at a sampling frequency of 5 kHz for approximately 50 minutes. *Note that while the different health conditions have no tangible impact on ECG-based authentication, we have included them to illustrate* *the robustness of our EBA system to different kinds of input signals.*

Template generation: As discussed in Section II-A, in the enrollment phase, the user's ECG signal is registered to generate the template. Based on the data sets, we were able to collect 40 different test samples (i.e., beats/segment) from any individual's ECG. In order to generate a template, we randomly selected 20 ECG beats (segments) from a total of 40 ECG beats and calculated the average to make a template for each individual user. We registered each user using one ECG segment, and each segment contains 24 feature sets that have been extracted in the feature extraction module. In other words, we generated template sets with sizes of $90 \times X$, $290 \times X$, and $20 \times X$ for the ECG-ID, PTB, and CEBSDB datasets, respectively. Here, X indicates the number of feature sets (24). After each user is registered in the template, we tested and evaluated our EBA system using 40 segments for each user.

Performance: We used all three datasets—ECG-ID database, PTB, and CEBSDB—to evaluate the EBA algorithm's matching task and recognition performance. In order to do that, we considered 40 ECG segments (beats) for each individual. Thus, we obtained test sets with sizes of 90×40 for the ECG-ID dataset, 290×40 for the PTB dataset, and 20×40 for the CEBSDB dataset. For each test sample, we calculated the Euclidean distance or similarity between the test sample and the template of each individual. To measure a genuine match score, a pair of samples from the same user have to be compared using the matching module; to measure an impostor match score, a pair of samples from two different users have to be compared. In the authentication phase, we took each subject as a genuine user and considered the rest as impostors. Therefore, we had a total of $Ns(s-1)/2$ genuine comparisons, and $N(N-1)s^2/2$ impostor comparisons, where N is a total user size and s is the number of segments (40). N is 90, 290, and 20 for the ECG-ID database, the PTB Diagnostic database, and the CEBSDB, respectively.

The test performances of the accuracy, *FAR*, *FRR*, and *EER* of the three datasets are shown in Table II and Fig. 5. As shown in Table II, the accuracy and *FRR* for all three datasets were 100% and 0%, respectively. In contrast, *FAR* for ECG-ID, PTB, and CEBSDB were 1.86%, 3.36%, and 3.5%, respectively, while the *EER* was 2%, 3%, and 3.6%, respectively.

Compared to other methods such as [21] and [22], our results are superior to the state-of-the-art methods with *EER* of 2%, while prior work achieved 4.46% and 10%, respectively. We can also observe that only a 93% accuracy is obtained from prior work [23] when the fiducial feature extraction has been implemented, while our proposed method achieves an accuracy of 100%.

III. Timing-Aware Domain-Specific Architecture for EBA

Even though there are a few prior works [3], [4], [5], [24] that proposed accelerators for EBA, ours is the first that explores the mitigation of side-channel attacks for the *whole* algorithm. Using a domain-specific architecture, as opposed to an ASIC or FPGA-based accelerator, offers the added benefit of flexibility to execute other applications on the base configuration. Furthermore, a domain-specific architecture lends itself to easier integration into general-purpose (or multipurpose) architectures used in consumer devices, such as smartphones and smartwatches [2].

TABLE I: The summary of the four data sets adopted in our experiments.

Dataset	Sample rate	# of subjects	Health status
ECG-ID	500	90	Healthy
PTB	1000	290	Healthy & Myocardial infarction
CEBSDB	5000	20	Healthy

TABLE II: A performance comparison of ECG biometric authentication using different datasets.

Dataset	# subjects	Accuracy	*FPR*	*FRR*	*EER*
ECG-ID	90	100	1.86	0	2
PTB	290	100	3.36	0	3
CEBSDB	20	100	3.5	0	3.6

Designing a domain-specific architecture for EBA involved a detailed and fine-grained analysis of the algorithm's execution characteristics and requirements. We followed three key guidelines in designing our architecture: (1) *using dedicated memories to minimize the distance of data movement*; (2) *eliminating unnecessary advanced arithmetic units and microarchitectural optimizations*; and (3) *using the easiest form of parallelism that matches the domain*. These guidelines, among others, are followed in the design of other domain-specific architectures [25]. In this section, we describe the different design decisions made in the architecture and the motivations for those decisions.

A. Leveraging Adaptability

Various steps in the ECG biometric algorithm have drastically different execution characteristics that require different resources for minimum or near-minimum latency. For example, whereas segmentation exhibited a high amount of instruction-level parallelism (ILP), we observed that the ILP in filtering was much less, and as such, did not benefit from out-of-order execution. Thus, our overall approach was to exploit our observations about the algorithm's execution characteristics to explore an architecture that is well-matched to the algorithm's needs. We further sought to use adaptability to ensure that the resources are *just enough* for each step, in order to minimize resource over-provisioning and achieve constant timing. For simplicity, we limited the employed adaptability to frequency scaling and shutting down the out-of-order backend when not in use (details in Section III-E).

B. Mitigating Timing-based Side-channel Attacks

EBA systems have been shown to have security and privacy issues in adversarial settings [26]. Major security threats in biometrics can be executed in three ways: *non-invasive*, *semi-invasive*, and *invasive*. Invasive attacks [27] are the most expensive and intrusive, involving physical tampering (e.g., circuit editing and micro-probing). Compared to invasive attacks,

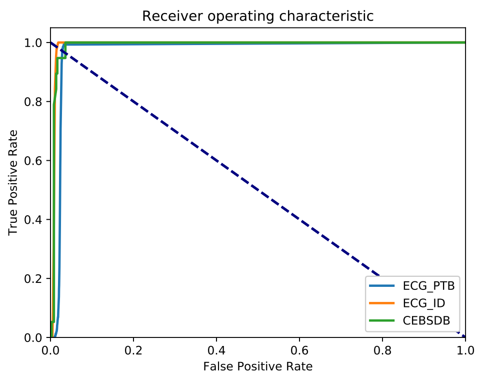


Fig. 5: ROC curves of ECG-based biometric authentication for ECG-ID, PTB, and CEBSDB datasets.

semi-invasive attacks [28] require moderate cost and some physical tampering (e.g., partial decapsulation and backside thinning). On the other hand, non-invasive attacks [29], such as side-channel attacks, require the lowest cost and no physical tampering. Side-channels are physical signatures/leakages from the execution time, power consumption, and electromagnetic emanations (EM) released while the device is manipulating data. Timing based side-channel leakage, within the context of EBA, occurs when an attacker obtains the execution time of the target authentication step (e.g., filtering, segmentation, feature extraction) for the provided input. The side-channel information can then be analyzed to infer private information. These side-channel attacks rely on the attacker's ability to detect the variability between different operations or inputs to the system [30].

In this paper, we focus on the timing leakage of each step of the EBA system based on their execution latency. Since different steps of the EBA take different amounts of time, attackers may infer the current steps of the authentication process, or even the specific instructions being executed, simply by monitoring the latency of executions. Therefore, our approach is to ameliorate the timing side-channels by ensuring that the timing information from latency profiles cannot be used to infer the operations being performed; thus, reducing the vulnerabilities in EBA computing. Our future work involves incorporating mitigation strategies for power attacks into the work proposed herein.

C. Threat Model

In this subsection, we define the threat model, as well as timing side-channel leaks under our threat model. In general, timing attacks are based on measuring how much time various computations in the EBA algorithm take to perform (e.g., comparing an attacker's given ECG signal with the victim's unknown signal). Given the classification of biometric technologies as a match-on card or match-on server (Section II), we define two types of attacks: *server-based attack* and *device-based attack*.

We assume that in a *server-based attack*, an adversary can observe the variation of the total execution time of the victims EBA algorithm with respect to the ECG signal. This capability is possible by accessing the match-on server that contains the database of all enrolled users' ECG signals. In this scenario, for example, the victims EBA runs on a server that can be remotely probed and timed by the attacker using malicious/fake ECG signals. Due to cross-matching problems and privacy invasion, an adversary can reconstruct various users ECG data through timing attacks. Mitigating such attacks at the server level will likely be very expensive. On the other hand, in *device-based attacks*, an adversary can directly access the victims device to observe variations in how long it takes to run the EBA algorithm. In this scenario, an attacker will only access the users ECG signal (only one user) that has been enrolled in the card (device).

In more detail, an attacker can access leaked information from the EBA system by measuring the time it takes to respond to certain queries (trying different ECG signals as inputs). Note that public ECG data is available to everyone and can be exploited by attackers to access confidential information leaked from timing attacks. Basically, an attacker attempts to compromise the victim's ECG signal by analyzing the time taken to execute the EBA algorithm. The different operations in our EBA algorithm take various amounts of time to execute, and the time can change based on the different inputs (ECG signals), given that each user's ECG signal produces a unique waveform. By precisely measuring the time for each operation, an adversary can work backward to the input and reconstruct the victims ECG data.

The importance of securing a biometric system against timing side-channel attacks is that the biometric data are permanent and not possible to revoke if compromised. To mitigate timing attacks, it is important to design EBA systems with constant-time functions and ensure that input-dependent timing variations are eliminated in the EBA system. The following subsection details how we implemented constant timing in the EBA algorithms as a countermeasure to timing attacks.

D. Input-Independent Intra-Step Constant Timing

Our overarching goal was to ensure constant timing across the different algorithm steps. However, we also observed that *intra-step* timing variations could occur due to different inputs to the algorithm (i.e., user data inputs). This intra-step timing variation was especially evident in the case of branches within the algorithm's instructions. Thus our first goal was to modify the code such that our EBA algorithm maintained constant timing regardless of inputs, similar to the *constant time exponentiation* [31], [32] mitigation technique for timing-based side-channel attacks.

To achieve this, we explored all the branches in our algorithm/code and inserted dummy computations (tantamount to *nops*) to balance the number of instructions in both the taken and not taken branches. That is, all branches execute the same number of instructions regardless of the branch direction. We found that the timing variations for different inputs were especially significant during the segmentation step for R-peak detection. This step consists of complex branches with computationally expensive loop operations and multiple

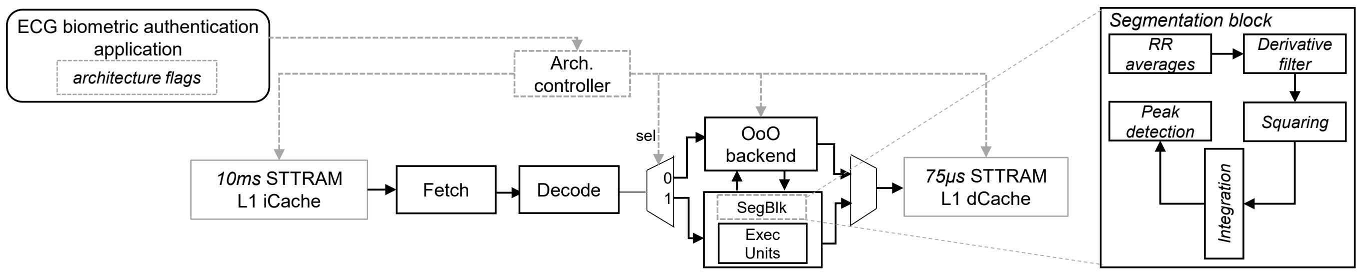


Fig. 6: Overview of domain-specific architecture (DSA) and computation flow in the custom segmentation block (*segblk*).

loop exit conditions. Simply balancing the full loops resulted in substantial execution time overhead due to the dummy computations. Thus, to mitigate this overhead, we observed that constant timing could be achieved by only inserting dummy computations equivalent to the maximum number of times those branches are called for any possible combination of inputs. The results of this intra-step constant timing process are presented in Section V.

E. EBA Domain-Specific Architecture (DSA)

Fig. 6 depicts a high-level overview of the proposed DSA. The architecture comprises of a base out-of-order core similar to modern-day ARM-based high-performance processors for consumer devices, such as smartphones. The functional units (execution units) include two ALUs, one load, and one store unit, while we eliminated complex functional units, such as the floating-point and single instruction multiple data (SIMD) units, as the EBA algorithm did not need them. Increasing the number of functional units did not provide any latency benefits and was deemed unnecessary.

For energy savings, we opted to use *spin-transfer torque RAM (STTRAM)* caches, given their low leakage power and normally-off computing capabilities [33]. We used a 4-way set associative 16KB cache with 64B blocks. A 32KB cache, which is common in smartphones, was over-provisioned for the algorithm. To further limit the energy overheads, we used reduced retention STTRAM caches that only retain data for a limited period of time, after which the data is invalidated [34]. Circuit-level details of how reduced retention can be implemented are outside the scope of this paper but have been described in prior work [34].

However, to prevent latency overheads or data corruption, prior work [35] has shown that the retention time must suffice for the cache blocks of the executing applications. Thus, we analyzed the cache blocks of the algorithm's different steps to reveal that the data cache blocks, on average, required approximately $75\mu s$, while the instruction cache blocks required $10ms$ on average. That is, most data and instruction cache blocks were either evicted or invalidated through normal cache accesses after $75\mu s$ and $10ms$, respectively. In general, instructions were more frequently reused, hence the longer retention time, while there was much higher dynamic data activity. Therefore, we used $75\mu s$ and $10ms$ retention times for the data and instruction caches, respectively. To prevent data corruption of blocks that need to remain in the cache beyond the retention time, we also incorporated a low-overhead 2-bit-per-block monitor counter, similar to prior work [35]. This counter, which is incorporated with the cache controller,

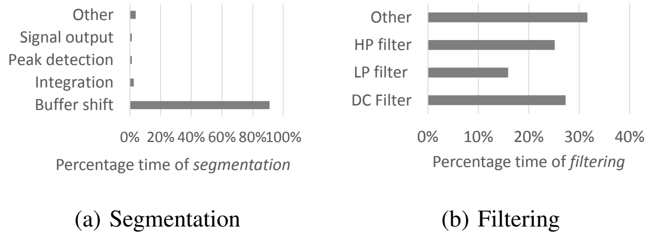


Fig. 7: Percentage time of various components of the filtering and segmentation steps. Time was relatively evenly spread in filtering, but the *buffer shift* loop in segmentation comprised the major source of overhead due to data movement to/from memory.

simply writes back and invalidates dirty cache blocks when the cache's retention time is about to elapse. Note that the use of STTRAM is orthogonal to the work proposed herein, and our work still achieves optimizations even with SRAM cache.

We observed that the segmentation step took the most execution time—98.6%—of the whole algorithm. Further fine-grained analysis revealed that the main latency bottleneck arose from the data movements between processor and memory in the segmentation step. Specifically, as illustrated in Fig. 7a, a single loop (*Buffer shift* in the figure) accounted for 91.02% of the overall execution time of segmentation. Comparatively, other steps' profiles were relatively stable throughout execution, as shown in Fig. 7b for filtering (for brevity, we omit figures for the other steps).

Therefore, to mitigate the bottleneck of the segmentation step, we explored two flavors of our domain-specific architecture. First, we designed a hardware implementation of a custom segmentation block (*segblk)*—effectively, an accelerator—to perform the required computations (Fig. 6). The *segblk* is tightly coupled to the core and shares key resources, such as the register file, memory management unit, and caches, with the core. While it imposes a less flexible integration than a loosely-coupled implementation, a tightly coupled *segblk* affords the benefit of zero runtime overhead for its invocation, which is vital for the EBA algorithm. The *segblk* has pointer-based inputs and outputs, and start and done signals, and communicates with the rest of the processor via the on-chip bus protocol. During the program execution, when the segmentation function is called, the start signal is given to the *segblk*. While the segmentation function is executed in the *segblk*, the processor can go to a low power state and wait for the execution to complete. Completion of the segmentation step triggers an interrupt signal to wake up the processor and continue the program execution.

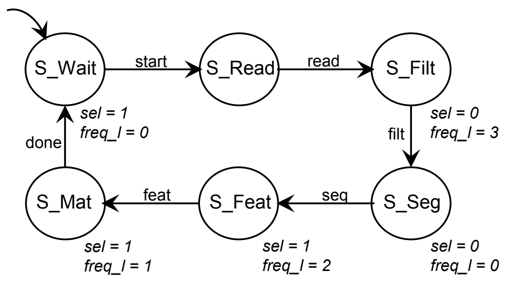


Fig. 8: Finite state machine of the architecture controller.

Fig. 6 illustrates the main functions implemented in the *segblk* and the dataflow among the functions. The *segblk* controller and datapath were implemented in Synthesizable Verilog to perform five major functions as follows: 1) initialization of *RR averages*; 2) *derivative filter* calculation using the Pan-Tompkins formula; 3) squaring the derivative to eliminate negative values and emphasize high frequencies; 4) moving-window integration; and 5) peak detection.

Second, as an alternative to the custom segmentation block, and given the memory bottleneck of the segmentation step, we introduced a 4KB dedicated buffer for the segmentation step. The buffer size was selected based on the amount of data movement observed during our analysis. This buffer, directly accessible via load/store instructions, reduces the distance and frequency of data movement. For low overhead, we used a STTRAM buffer with a $100\mu s$ retention time (SRAM can also be used with similar results, but higher energy overheads). The tradeoffs and overheads of these architectures are discussed in Section V.

Finally, to enforce constant timing, our DSA adapts the architecture to different algorithm steps. Adaptability is achieved through dynamic frequency scaling [10], which is commonly implemented in modern-day processors, and by varying the execution order (i.e., in-order vs. out-of-order). To vary the execution order, instructions can be multiplexed through the execution pipeline in an in-order fashion or through the out-of-order backend, depending on the algorithm step. The architecture is implemented as illustrated in Fig. 6, and is conceptually similar to the *composite core* architecture [11]. All instructions traverse through the fetch and decode stages of the pipeline as usual. However, depending on the multiplexer's select signal, as dictated by the *architecture controller*, the instructions can traverse through the out-of-order backend (when sel = 0) or through the execute stage of the pipeline in program order (when sel = 1).

Fig. 8 depicts the controller's state machine. For clarity, only the most important signals shown. Output signals are shown in italics and don't change unless specified within a state. We implemented the controller as a simple six-state finite state machine (FSM) with different states for each of the four steps of the algorithm (*S_Filt, S_Seg, S_Feat*, and *S_Mat*), a state for reading the signals (*S_Read*), and the initial 'wait' state (*S_Wait*). The inputs to the FSM, which trigger its state transitions, are single-bit *architecture flags* that are asserted at the completion of each EBA step. That is, for example, at the end of the filtering step, the *filt* signal is asserted, at the end of the segmentation step, the *seg* filter is asserted, and so on. The architecture flags are added at design time to the EBA code. Each FSM state outputs the necessary signals to configure the architecture for each algorithm step, as depicted in Table III.

TABLE III: Constant-timing configuration for each step. *Freq_l* and *Sel* represent the frequency level ID and select signal output in the architecture controller.

EBA Step	Frequency	Freq_l	Execution	Sel
Filtering	600 MHz	3	In-order	0
Segmentation	2.1 GHz (base)	0	Out-of-order	1
Feature extraction	500 MHz	2	In-order	0
Matching	400 MHz	1	In-order	0

To determine the appropriate configuration to achieve constant timing for each step, we used a simple design-time heuristic. The heuristic uses a greedy strategy to perform an interleaved exploration of the execution order and clock frequency in order to determine which configurations achieve timing within $500\mu s$ of the segmentation step. We used $500\mu s$ as our threshold since timing variations of such small magnitude are generally undetectable by side-channel attackers [31], [36]. Our exploration heuristic occurred as follows: starting from the base out-of-order configuration, we explored each frequency for each step of the EBA algorithm in descending order, and then similarly for the in-order configuration. Exploration continued until the execution time was within $500\mu s$ of the segmentation step. The base clock frequency was 2.1 GHz, and the frequencies were explored in decrements of 100 MHz. The timing was compared to the segmentation step since it was the most resource-demanding step and took the longest time. As such, other steps' execution had to be elongated to match the segmentation step's execution. Table III depicts the specific configurations selected by our heuristic for the different steps.

While this proposed architecture suffices for our EBA algorithm, we acknowledge that one limitation of this architecture is that additional design space exploration may be necessary to maintain constant timing if the algorithm is modified.

IV. Experimental Setup

We implemented the EBA algorithm in C with flags to demarcate the various steps described in Section II. The original code[1] was also modified to achieve the intra-step timing, as described in Section III-D, and to provide inputs to the controller (Section III-E). We cross-compiled the code for the ARM instruction set architecture (ISA) for our experiments.

For the baseline processor architecture, we used configurations similar to the ARM Cortex A15. The processor features a 2100 MHz base clock frequency, separate 16KB L1 instruction and data caches, and an 8GB main memory. To provide a fair comparison, for the baseline processor, we also assumed reduced retention STTRAM caches—$75\mu s$ and $10ms$ for the instruction and data caches, respectively. However, we also compare our work with a generic processor featuring SRAM to provide a robust evaluation of our work. To model the domain-specific architectures (DSA) proposed herein and gather execution statistics of the ECG algorithm, we used an

[1]The code is available at: www.ece.arizona.edu/tosiron/downloads.php

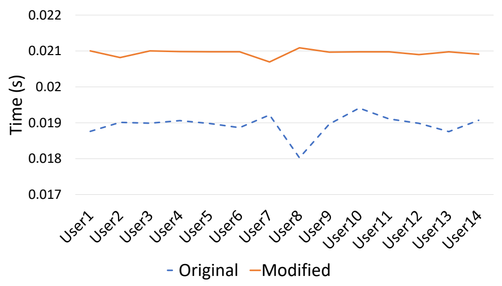


Fig. 9: Runtime of original and modified EBA code. For brevity, the graph shows a subset of user inputs is shown.

in-house modified version of the GEM5 simulator [37], which allows us to model STTRAM caches with reduced retention times. To model the power and energy, we used a combination of NVSim [38] and McPAT [39] integrated with the GEM5 statistics. To model the STTRAM caches, we used the MTJ cell modeling technique proposed in [40] to obtain design parameters, such as the write pulse, write current, and the resistance value R_{AP}, and applied these parameters to NVSim. We implemented the custom segmentation block ($segblk$) using synthesizable Verilog and Xilinx Vivado synthesis.

V. Experimental Results

In this section, we evaluate the latency and energy savings of the proposed architecture compared to the base architecture. We also evaluate the ability of the adaptable DSA to achieve constant *and* reduced timing for the EBA algorithm, as compared to the base. We evaluate three different versions of the DSA: with the custom block (DSA_segblk), with the dedicated segmentation buffer (DSA_buff), and with buffer and adaptability for constant timing (DSA_adapt). The specific configurations used for the different steps of our ECG algorithm in the adaptable DSA are shown in Table III. We also evaluate the overhead accrued by the proposed architecture and compare our architecture to prior work.

A. Timing Analysis

Intra-step input-independent constant timing: First, we explore the ability of our modified EBA code to achieve constant timing for different user inputs as described in III-D. Fig. 9 presents the runtime simulations of the EBA algorithm with various inputs for the original and modified code. These experiments were performed using the baseline processor configuration.

As seen in the figure, there were timing variations among the different user inputs, whereas the modified code kept the timing relatively constant. In the original code, the maximum timing variation due to input changes was 1.4ms, while the maximum variation for the modified code was 400μs (less than the 500μs threshold). Overall, the modified code increased the latency compared to the original code by 11.17% on average. This increase in latency resulted mainly from the increase in the number of instructions in the modified code (by about 4.19%), due to the balancing of branches (Section III-D).

In what follows, we analyze the latency and energy of our approach on different versions of our domain-specific architecture.

Latency and inter-step constant timing: Despite the execution time increase from the software changes, our domain-specific architecture still achieved substantial latency improvements compared to the base. We first present the comparisons of the DSA to the base with the modified code running on all the systems.

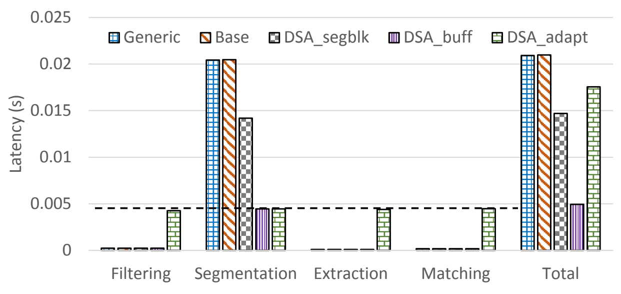
Fig. 10(a) depicts the latency (in s) of the different versions of our DSA and the base architecture for the different algorithm steps. On the base configuration, the execution times of the different steps of the ECG algorithm were widely disparate. For instance, the matching step took 1.77x more time than the feature extraction, the filtering step took 1.37x more time than the matching step, and the segmentation step took 87.88x more time than the filtering step. (Due to the different time scales, these differences are not clearly visible in Fig. 10(a)).

Introducing the custom block for the segmentation step (DSA_segblk) reduced the latency of the segmentation step by 44% and reduced the total latency (for the whole algorithm) by 42%. Interestingly, since segmentation was memory-bound, DSA_buff (DSA with dedicated segmentation buffer) achieved more performance benefits than DSA_segblk. DSA_buff achieved the highest latency improvement of 4.24x compared to the base, and 2.97x improvement compared to DSA_segblk. Note that while the code exhibited constant intra-step timing, DSA_segblk and DSA_buff did not keep the timing constant across the different steps; that wasn't an optimization goal for these architectures as in DSA_adapt.

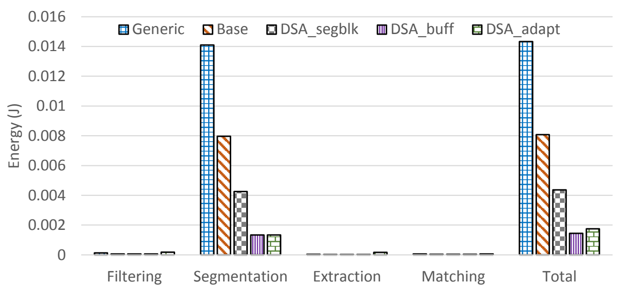
As intended, DSA_adapt achieved constant timing (dotted line in Fig. 10) across the different steps. Furthermore, given software changes, both the intra-step and inter-step timing variations were mitigated. However, even though DSA_adapt improved the latency compared to the base architecture, the constant timing was achieved at the expense of latency optimization compared to DSA_segblk and DSA_buff.

DSA_adapt reduced the latency compared to the base by 19%, but *increased* the latency by 19% and 3.55x compared to DSA_segblk and DSA_buff, respectively. This tradeoff occurred in DSA_adapt because in order to achieve constant timing, other steps' timing had to be increased to match the minimum possible timing achieved for segmentation. Similarly to prior research [30], [31], by achieving constant timing across the different distinct steps of the algorithm, DSA_adapt mitigates the chances of an attacker being able to obtain patterns in the timing profile to perform a timing-based side-channel attack.

Compared to a generic ARM architecture with SRAM (rather than STTRAM as in our DSA), DSA_segblk, DSA_buff, and DSA_adapt improved the latency by 1.42x, 4.23x, and by 19%, respectively. These latency improvements were similar to those achieved by the base architecture featuring STTRAM (the STTRAM architecture increased the latency by 0.3%). Minimizing the latency overhead compared to SRAM was possible as a result of the a priori analysis of the EBA algorithm's cache block requirements in order to determine the retention time that satisfies the needs of the



(a)



(b)

Fig. 10: (a) Timing and (b) Energy of algorithm steps on the different versions of the *(DSA)* compared to the base (*Base*).

algorithm's cache blocks.

Modified code on the DSA vs. the original code on the base: To provide a complete picture of the results, and to further illustrate the robustness of our work, we also compared DSA_segblk, DSA_buff, and DSA_adapt while running the modified code (with the increased instruction count) to the base while running the original code. As shown in Fig. 11, compared to the base running the original code, DSA_segblk, DSA_buff, and DSA_adapt reduced the latency by 29.26%, 3.85x, and 8.22%, respectively.

Comparison to prior work: To evaluate our work in the context of prior work, we compared DSA_adapt to the implementation of Ozone [30] for our EBA algorithm. Ozone is a hardware technique targeted at mitigating timing-based side-channel attacks by ensuring that applications execute with a fixed latency regardless of inputs. Ozone achieved similar results to our work in ensuring constant timing, albeit with different tradeoffs for both techniques. As depicted in Fig. 11, Ozone accrued execution time overhead and increased the latency by 4.55x, whereas DSA_adapt *reduced* the latency by 8.22% compared to the base architecture (running the original code). However, an advantage of Ozone over our work is that it would achieve constant timing for a variety of applications, whereas DSA_adapt would require a priori knowledge of the application and design-time modifications to enable constant timing. Our work had the benefit of a priori application knowledge and profiling to modify the code such that the latency overhead was minimized. We believe that a synergy of DSA_adapt and a technique like Ozone would be beneficial to mitigate the drawbacks of both techniques, and we plan to explore this synergy in future work.

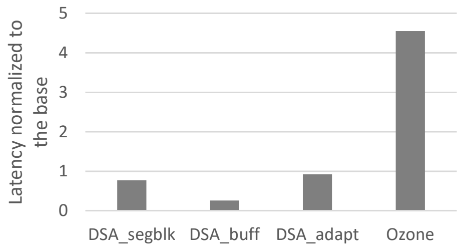


Fig. 11: Latency of DSA designs (DSA_segblk, DSA_buff, and DSA_adapt) and prior work (*Ozone* [30]) normalized to the base (baseline of 1). The base is running the original code while the other architectures are running modified codes.

B. Energy Consumption

Fig. 10(b) depicts the energy consumed—comprising of both static and dynamic energy—by the different versions of our DSA compared to the base (running the modified code). Compared to the base, DSA_segblk, DSA_buff, and DSA_adapt reduced the total energy by 1.85x, 5.59x, and 4.62x, respectively. Compared to the base running the original code, DSA_segblk, DSA_buff, and DSA_adapt reduced the energy by 67.15%, 5.06x, and 4.18x. The majority of the energy saving was achieved by introducing the segmentation buffer in DSA_buff, and DSA_adapt. Despite the increase in latency on DSA_adapt, substantial energy savings was achieved due to frequency scaling and shutting down the out-of-order backend for all the steps other than segmentation.

While the buffer was a source of some power overhead, its significant impact on the latency resulted in energy savings. Similarly, the $segblk$ also reduced the energy compared to the base despite its power consumption. Furthermore, even though DSA_adapt increased the energy consumption of the other steps (in order to achieve constant timing), by mitigating the bottleneck of the segmentation step, the architecture still achieved significant overall energy savings. Compared to the generic ARM processor with SRAM caches, incorporating STTRAM with a specialized retention time and eliminating the FP and SIMD units achieved significant energy benefits. The base DSA reduced the energy by 1.77x, while DSA_segblk, DSA_buff, and DSA_adapt reduced the energy by 3.27x, 9.92x, and 8.20x, respectively.

C. Evaluation with a New EBA Algorithm

To further evaluate the robustness of the DSA, we also quantified the latency and energy benefits while running a new EBA algorithm. We used an EBA algorithm similar to [23] that featured a discrete wavelet transform for feature extraction. Fig. 12 depicts the latency and energy of DSA_buff, and DSA_adapt normalized to the base architecture. For brevity, only a summary of the overall latency and energy results are shown. Compared to the base, DSA_buff reduced the latency and energy by 75.43% and 81.58%, respectively, while DSA_adapt reduced the latency and energy by 17.14% and

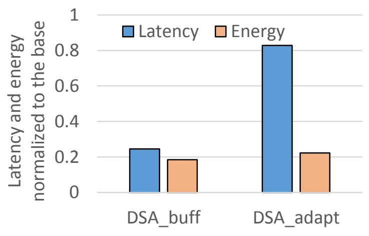


Fig. 12: Latency and energy of DSA designs (DSA_buff, and DSA_adapt) normalized to the base (baseline of 1) while running a new EBA algorithm.

77.77%, respectively. Notably, these results were achieved without any design modifications to the DSA. Furthermore, DSA_adapt achieved a constant timing for the new algorithm with some code modification and design space exploration (see Section III), but the DSA remained unchanged.

D. Overhead

The main overheads of our work result from the dedicated segmentation buffer, controller, and the custom segmentation block, *segblk* (Section III-E, Fig. 6). The buffer resulted in 0.002mm^2 and 1.97mW area and power overheads, respectively, and had read and write latencies of 0.250ns and 0.988ns, respectively. We also implemented the controller (Section III-E) using synthesizable Verilog, and estimated that the overheads were negligible: the controller's critical path was only 4.24ns; the area and power were approximately 0.001mm^2 and 3mW, respectively. Finally, we implemented the segmentation custom block ($segblk$) on a Zynq-7000 FPGA to evaluate its overheads. The power overhead was 0.3W, and 14829 LUTs were used for the design.

VI. Related Work

In [41], attacks were performed on an FPGA-based convolutional neural network accelerator to recover the input image from the collected power traces without knowing the detailed parameters in the neural network. Their investigations resulted in the reconstruction of digit images of the MNIST dataset. Another similar work was presented by [42], where they reverse-engineered a neural network (multilayer perceptron) by using non-invasive power side-channel leakage information. Their method is able to recover secret inputs from a known network with only a single-shot side-channel analysis. In [43], the authors showed how a Neural Network model is susceptible to timing side-channel attacks. They proposed a black box Neural Network extraction attack by exploiting the timing side-channels to infer the depth of the network.

To the best of our knowledge, our work is the first attempt to explore countermeasures to side-channel attacks in EBA systems. The most related work to ours in this respect is [7], where the authors investigated the vulnerability of stored features in fingerprint biometric authentication to side-channel attacks. They presented SPA-based side-channel attacks on fingerprint matching algorithms. Other prior works [30], [42] have studied the susceptibility of systems to timing side-channel attacks and shown the benefits of maintaining constant timing as a countermeasure against such attacks.

There have also been prior works that employ hardware accelerators for EBA. Page *et al.* [4] proposed a 307-node hidden layer deep neural network design implemented on an FPGA for EBA targeting embedded systems. Yin *et al.* [3] proposed a 65-nm processor that performs real-time biometric authentication as well as personal cardiac monitoring. Kang *et al.* [24] proposed an ECG authentication system design for mobile and wearable devices using the ARM Cortex-M processor for their design. In comparison to these prior works, our work offers the important advantage that it mitigates timing-based side-channel attacks on EBA systems, while also allowing the flexibility of executing other applications when the authentication algorithm is not running.

VII. Conclusion and Future Work

In this paper, we proposed a domain-specific architecture (DSA) for an ECG biometric authentication (EBA) system. We explored multiple versions of the architecture, one of which uses adaptability to trade off latency and energy minimization for constant timing across all steps of the authentication algorithm. Thus, this architecture mitigates the EBA system's vulnerability to timing-based side-channel attacks. The proposed architectures substantially reduce the latency and energy of the EBA algorithm compared to a base ARM-based processor architecture. Our work represents an important step towards domain-specific architectures for secure EBA systems. However, studies of side-channel attack in EBA systems is still nascent. For future work, we plan to expand the architecture proposed herein to also ensure a constant power profile in order to mitigate power side-channel attacks. Additionally, we intend to explore further opportunities for reducing the latency and energy tradeoffs of the adaptable architecture.

References

[1] I. Odinaka, P.-H. Lai, A. D. Kaplan, J. A. O'Sullivan, E. J. Sirevaag, and J. W. Rohrbaugh, "Ecg biometric recognition: A comparative analysis," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1812–1824, 2012.

[2] W. Meng, D. S. Wong, S. Furnell, and J. Zhou, "Surveying the development of biometric user authentication on mobile phones," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1268–1293, 2014.

[3] S. Yin, M. Kim, D. Kadetotad, Y. Liu, C. Bae, S. J. Kim, Y. Cao, and J.-s. Seo, "A 1.06-μw smart ecg processor in 65-nm cmos for real-time biometric authentication and personal cardiac monitoring," *IEEE Journal of Solid-State Circuits*, 2019.

[4] A. Page, A. Kulkarni, and T. Mohsenin, "Utilizing deep neural nets for an embedded ecg-based biometric authentication system," in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2015, pp. 1–4.

[5] S. K. Cherupally, G. Srivastava, S. Yin, D. Kadetotad, C. Bae, S. J. Kim, and J.-s. Seo, "Ecg authentication neural network hardware design with collective optimization of low precision and structured compression," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[6] M. Borowczak, "Side channel attack resistance: Migrating towards high level methods," Ph.D. dissertation, University of Cincinnati, 2013.

[7] M. Dürmuth, D. Oswald, and N. Pastewka, "Side-channel attacks on fingerprint matching algorithms," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '16. New York, NY, USA: ACM, 2016, pp. 3–13.

[8] I. Martinovic, D. Davies, M. Frank, D. Perito, T. Ros, and D. Song, "On the feasibility of side-channel attacks with brain-computer interfaces," in *Presented as part of the 21st {USENIX} security symposium ({USENIX} Security 12)*, 2012, pp. 143–158.

[9] S. Eberz, G. Lovisotto, A. Patane, M. Kwiatkowska, V. Lenders, and I. Martinovic, "When your fitness tracker betrays you: Quantifying the predictability of biometric features across contexts," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 889–905.

[10] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED'07)*. IEEE, 2007, pp. 38–43.

[11] A. Lukefahr, S. Padmanabha, R. Das, F. M. Sleiman, R. Dreslinski, T. F. Wenisch, and S. Mahlke, "Composite cores: Pushing heterogeneity into a core," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2012, pp. 317–328.

[12] N. Karimian, P. A. Wortman, and F. Tehranipoor, "Evolving authentication design considerations for the internet of biometric things (iobt)," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2016, pp. 1–10.

[13] F. Agrafioti, K. Martin, and S. Oung, "Preauthorized wearable biometric device, system and method for use thereof," Mar. 31 2015, uS Patent 8,994,498.

[14] A. Lourenço, A. P. Alves, C. Carreiras, R. P. Duarte, and A. Fred, "Cardiowheel: Ecg biometrics on the steering wheel," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 267–270.

[15] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, March 1985.

[16] J. R. Pinto, J. S. Cardoso, and A. Lourenço, "Evolution, current challenges, and future possibilities in ecg biometrics," *IEEE Access*, vol. 6, pp. 34 746–34 776, 2018.

[17] A. K. Jain, A. A. Ross, and K. Nandakumar, *Introduction to biometrics*. Springer Science & Business Media, 2011.

[18] T. Lugovaya, "Biometric human identification based on electrocardiogram," *Master's thesis, Faculty of Computing Technologies and Informatics, Electrotechnical University LETI, Saint-Petersburg, Russian Federation*, 2005.

[19] M. A. García-González, A. Argelagós-Palau, M. Fernández-Chimeno, and J. Ramos-Castro, "A comparison of heartbeat detectors for the seismocardiogram," in *Computing in Cardiology 2013*. IEEE, 2013, pp. 461–464.

[20] R. Bousseljot, D. Kreiseler, and A. Schnabel, "Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet," *Biomedizinische Technik/Biomedical Engineering*, vol. 40, no. s1, pp. 317–318, 1995.

[21] H.-S. Choi, B. Lee, and S. Yoon, "Biometric authentication using noisy electrocardiograms acquired by mobile sensors," *IEEE Access*, vol. 4, pp. 1266–1273, 2016.

[22] S. Wahabi, S. Pouryayevali, S. Hari, and D. Hatzinakos, "On evaluating ecg biometric systems: Session-dependence and body posture," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 2002–2013, 2014.

[23] J. Liu, L. Yin, C. He, B. Wen, X. Hong, and Y. Li, "A multiscale autoregressive model-based electrocardiogram identification method," *IEEE Access*, vol. 6, pp. 18 251–18 263, 2018.

[24] S. J. Kang, S. Y. Lee, H. I. Cho, and H. Park, "Ecg authentication system design based on signal analysis in mobile and wearable devices," *IEEE Signal Processing Letters*, vol. 23, no. 6, pp. 805–808, 2016.

[25] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018.

[26] N. Karimian, D. Woodard, and D. Forte, "Ecg biometric: Spoofing and countermeasures," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 3, pp. 257–270, 2020.

[27] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 95–112, 2015.

[28] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *2004 International Conference on Test*. IEEE, 2004, pp. 339–344.

[29] S. Faezi, S. R. Chhetri, A. V. Malawade, J. C. Chaput, W. H. Grover, P. Brisk, and M. A. Al Faruque, "Oligo-snoop: A non-invasive side channel attack against dna synthesis machines." in *The Network and Distributed System Security Symposium (NDSS)*, 2019.

[30] Z. B. Aweke and T. Austin, "Øzone: Efficient execution with zero timing leakage for modern microarchitectures," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1123–1128.

[31] Y. Yarom and K. Falkner, "Flush+reload: A high resolution, low noise, l3 cache side-channel attack," in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 719–732.

[32] O. Aciiçmez, c. K. Koç, and J.-P. Seifert, "On the power of simple branch prediction analysis," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS 07. New York, NY, USA: Association for Computing Machinery, 2007, p. 312320.

[33] T. Kawahara, "Scalable spin-transfer torque ram technology for normally-off computing," *IEEE Design & Test of Computers*, no. 1, pp. 52–63, 2010.

[34] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, 2011, pp. 50–61.

[35] K. Kuan and T. Adegbija, "Lars: Logically adaptable retention time stt-ram cache for embedded systems," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 461–466.

[36] M. Schwarz, C. Maurice, D. Gruss, and S. Mangard, "Fantastic timers and where to find them: High-resolution microarchitectural attacks in javascript," in *Financial Cryptography and Data Security*, A. Kiayias, Ed. Springer International Publishing, 2017, pp. 247–267.

[37] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[38] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[39] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 469–480.

[40] K. C. Chun, H. Zhao, J. D. Harms, T.-H. Kim, J.-P. Wang, and C. H. Kim, "A scaling roadmap and performance evaluation of in-plane and perpendicular mtj based stt-mrams for high-density cache memory," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 2, pp. 598–610, 2012.

[41] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM, 2018, pp. 393–406.

[42] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 515–532.

[43] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *CoRR*, vol. abs/1812.11720, 2018. [Online]. Available: http://arxiv.org/abs/1812.11720