FuzzyKey: Comparing Fuzzy Cryptographic Primitives on Resource-Constrained Devices

Mo Zhang[1,4], Eduard Marin[2], David Oswald[1], and Dave Singelée[3]

[1] University of Birmingham, UK, mxz819@cs.bham.ac.uk, d.f.oswald@bham.ac.uk
[2] Telefonica Research, Spain, eduard.marinfabregas@telefonica.com
[3] imec-COSIC, KU Leuven, Belgium, dave.singelee@esat.kuleuven.be
[4] University of Melbourne, Australia

Abstract. Implantable medical devices, sensors and wearables are widely deployed today. However, establishing a secure wireless communication channel to these devices is a major challenge, amongst others due to the constraints on energy consumption and the need to obtain immediate access in emergencies. To address this issue, researchers have proposed various key agreement protocols based on the measurement of physiological signals such as a person's heart signal. At the core of such protocols are fuzzy cryptographic primitives that allow to agree on a shared secret based on several simultaneous, noisy measurements of the same signal. So far, although many fuzzy primitives have been proposed, there is no comprehensive evaluation and comparison yet of the overhead that such methods incur on resource-constrained embedded devices. In this paper, we study the feasibility of six types of fuzzy cryptographic primitives on embedded devices for 128-bit key agreement. We configure several variants for each fuzzy primitive under different parameter selections and mismatch rates of the physiological signal measurements on an MSP430 microcontroller, and then measure and compare their energy consumption and communication overhead. The most efficient constructions consume between 0.021 mJ and 0.198 mJ for the transmitter and between 0.029 mJ and 0.380 mJ for the receiver under different mismatch rates. Subsequently, we modify the best performing methods so that they run in constant time to protect against timing side-channel attacks, and observe that these changes only minimally affect resource consumption. Finally, we provide open-source implementations and energy consumption data of each fuzzy primitive as a reference for real-world designs.

Keywords: fuzzy commitment · fuzzy vault · fuzzy extractor · physiological signal · key agreement · energy consumption

1 Introduction

Healthcare technology is evolving at a rapid pace. Medical sensors are getting more miniaturised, while being able to measure a broader set of people's Physiological Signals (PSs) more reliably. New generations of widely-deployed Implantable Medical Devices (IMDs) are considerably lighter and smaller compared to previous generations. Wearables are extensively used nowadays, also

often within the context of health monitoring. Multiple wearable and medical computing devices can be connected to form a body area network. Besides their application opportunities within the health domain, all these devices have in common that they rely on a wireless interface to communicate with each other or with external devices such as a smartphone. This increased wireless connectivity enhances without any doubt the quality of the (remote) healthcare that can be offered to users. However, in turn, security and privacy are at stake for such medical systems. The medical data that is being monitored on the user is clearly privacy-sensitive. Moreover, the integrity and authenticity of the data, as well as remote updates or commands sent to the devices, have to be protected as well. Unfortunately, researchers have demonstrated that several medical and wearable devices available on the market currently lack security mechanisms [10,21–23,28].

It is therefore evident that cryptographic solutions are needed to secure the wireless interface between these devices. This includes the initial security bootstrap process to establish a secret session key to protect the wireless communication link. However, this turns out to be a challenging research problem for various reasons. First, most of these devices have strict resource constraints, e.g., limited memory and computational power. Furthermore, most IMDs are operated by a single non-rechargeable and non-replaceable battery which typically lasts between five and seven years (depending on the type of device and treatment). Once the battery is drained, the IMD is replaced through a surgical intervention that can pose risks to patients. Likewise, wearables typically contain small batteries, e.g., powered by a button cell with approximate a thousand joules. Thus, in such resource-constrained devices every single joule matters. Second, these devices often do not have any input or output interfaces, such as a keypad or a screen. Third, a subset of these medical devices, more particularly IMDs, are not even physically accessible at all, because they are implanted in the patient. Fourth, most of the wireless connections that have to be made with these devices cannot rely on any prior trust relation. This is because these network connections are not static, *i.e.*, the set of external devices one needs to connect to can change quite often. For example, during an emergency situation, the first doctor that is present (who may have never seen the patient before) might have to establish a communication link to the patient's IMD. Due to all these constraints, conventional key distribution and bootstrap techniques are not viable options: key exchanges based on public key cryptography are difficult to manage because they require establishment of a robust Public Key Infrastructure (PKI).

The use of physiological signals (e.g., a signal extracted from the user's heartbeat) has been proposed as an alternative to securely establish a key between two devices that do not have any prior trust relationship. In contrast to biometrics, where the extracted information is to some extent invariant, PSs are required to be random signals that vary over time. The security of PS-based cryptographic solutions relies on the fact that the user's PS can *only* be obtained by making physical contact with them (e.g., by touching the skin long enough). A common approach to agree on a key is for each of the devices to independently and synchronously take a measurement of a given user's PS [19, 20]. However, the

measurements taken by the devices are often not identical but at best rather similar due to inherent noise introduced by the measuring process. To address this limitation, Juels et al. [14,15] and Dodis et al. [9] introduced so-called *fuzzy cryptographic primitives*, including the fuzzy commitment [15], fuzzy vault [14] and fuzzy extractor [9], which allow two devices to agree a cryptographic key from noisy data[5].

1.1 Related Work

Fuzzy cryptographic primitives have become the basis of several PS-based cryptographic protocols. For example, K Venkatasubramanian et al. [37], Hu et al. [13] and Reshan et al. [3] utilised the fuzzy vault for key agreement based on measurements of InterPulse Intervals (IPIs), *i.e.*, time intervals between R-peaks of Electrocardiogram (ECG) signal. Similarly, Cherukuri et al. proposed a PS-based key distribution protocol based on the fuzzy commitment that is used to securely transport a session key between two sensors [6]. Another example is the key agreement protocol by Marin et al. which uses a fuzzy extractor in combination with IPIs [20]. It is worth noting that the security of PS-based key exchange protocols has been exhaustively investigated over the past years. In particular, Calleja et al. [5] and Seepers et al. [31] demonstrated that some PS, such as those extracted from the patient's heart, might be measured remotely without the need for direct physical touch. Besides, the entropy of the PS itself has been questioned, e.g., although IPI was frequently chosen as the PS used in prior security protocols, Ortiz-Martin et al. [25] challenged that IPIs may not have as much entropy as expected. Furthermore, some PS-based key exchange protocols, such as [6, 29], have been proved to be vulnerable to certain attacks [19].

While PS-based solutions have been frequently designed and analysed, little effort has been devoted into studying the feasibility of fuzzy primitives (as the core of such schemes) in resource-constrained systems as well as how to configure them to optimise performance. This is in contrast to "traditional" cryptographic algorithms, whose efficient implementation on resource-constrained devices has been widely studied, see e.g., [11, 24, 32].

1.2 Contributions

In this paper, we present implementations and evaluations of PS-based 128-bit key agreement based on fuzzy cryptographic primitives on an MSP430, which is a representative low-power microcontroller similar to the one used in commercial IMDs or wearables. Our main contributions are:

1. We implement and optimise six fuzzy cryptographic primitives for PS-based key exchange. Our implementation can be easily ported to different platforms.

[5] Apart from being used in PS-based key exchange protocols, fuzzy schemes are also used in other areas such as biometrics and Physical Unclonable Functions (PUFs) [2, 4, 8, 9], where traditional cryptographic algorithms are not directly applicable.

2. We evaluate and compare the resource consumption (energy consumption and communication overhead) of each construction under various parameter settings both at the transmitter and receiver using an MSP430. We demonstrate that fuzzy primitives are feasible on a resource-constrained embedded device. We show how parameter selection affects the performance and report on the overall best-performing fuzzy primitives under different metric spaces. To the best of our knowledge, we are the first to provide a systematic evaluations of various fuzzy primitives on resource-constrained devices.
3. We implement countermeasures against timing attacks for the most efficient constructions, and show that our protected implementations reduce timing leakage below the statistical significance threshold, while only minimally affecting resource consumption.

Our source code is available under the following link: https://github.com/MrZMN/FuzzyKey

Paper organisation. The remainder of this paper is organised as follows: in Section 2, we introduce the mathematical background of fuzzy primitives, commonly used components, and concrete constructions used in this paper. In Section 3, we explain our security assumptions and how to instantiate the constructions of fuzzy primitives. We give implementation details in Section 4, before evaluating the performance of all fuzzy primitives in Section 5. We conclude in Section 6.

2 Background

In this section, we describe the mathematical background required for this paper, and discuss several fuzzy cryptographic algorithms. Elementary computations are in $GF(2^m)$. In this paper, we consider $m \leq 8$ so that computations are fast on constrained embedded devices and most variables fit in one byte. A metric space M is a finite set. For each M, there is a definite integer distance $dist\,(m_1, m_2)$ between any two elements m_1 and m_2. The fuzzy primitives discussed in this paper rely on two different kinds of metric spaces: (i) *Hamming metric space* and (ii) *set metric space*. In a Hamming metric space, $M = F^\ell$ for an alphabet F. $dist(m_1, m_2)$ in M is the Hamming distance, which is the number of positions that m_1 differs from m_2. For example, for $M = \{0, 1\}^3$, $dist(\{0, 0, 0\}, \{1, 1, 1\}) = 3$. Besides, the number of non-zero elements in m_1 is called m_1's Hamming weight. In a set metric space, M contains all s-element subsets of a universe U. $dist()$ in M is the set difference, which is the size of symmetric difference (defined by $symdiff(m_1, m_2) = \{x \in m_1 \cup m_2 \mid x \notin m_1 \cap m_2\}$). For example, $U = GF(2^3)$ and $s = 3$, $dist(\{0, 1, 2\}, \{0, 1, 3\}) = 2$. $dist(m_1, m_2)$ is an even number when the size of m_1 and m_2 is the same.

The inputs of the fuzzy primitives working on these two metrics are different. For this paper, the input is the physiological value converted from a PS. For Hamming metric methods, the input is a bit string that can be generated by concatenating the bit representations of PSs, such as heart rates, which makes these methods flexible for different types of PS. However, because a bit string is

consecutive, these methods are sensitive to dislocation and erasure errors on the measurements (e.g., due to peak misdetection when using the heart beat [30]). One bit erasure at the start of a bit string might lead to a significant increase in the Hamming distance. Set metric methods alleviate these problems to a certain extent. The input in this case is a set, and even if there are order-difference or erasure problems on set elements, the set difference will not vary substantially. However, converting one specific PS into a set whose elements are randomly distributed in U can be complicated, especially when the size of U is large.

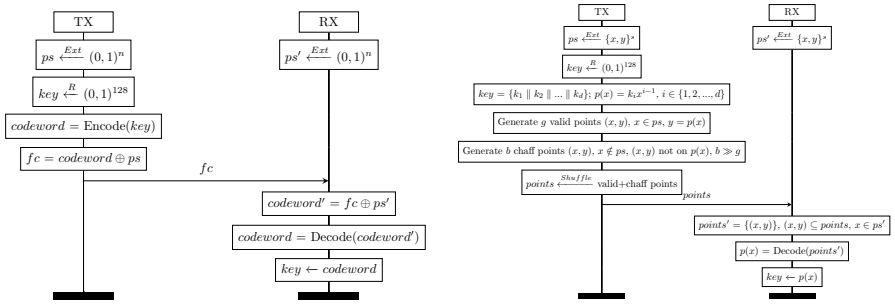
Error correction codes. Error Correction Codes (ECCs) are frequently used to achieve error-tolerance in this paper. An ECC comprises *encoding* and *decoding* phases. In the encoding phase, the original data is encoded as a codeword, where some form of redundancy is added. When errors appear in the codeword, the decoding phase recovers the original data if the total number of errors is below the error tolerance limit. ECCs are represented by the triple $\{n, k, t\}$, where n is the number of symbols of the codeword, k is the number of symbols of the data ($k < n$), and t is the maximum number of errors that can be corrected in a codeword. The error tolerance is then t/n. We focus on two linear ECCs, namely binary Bose-Chaudhuri-Hocquenghem (BCH) and Reed-Solomon (RS) codes, as already recommended in the first papers on fuzzy primitives [9, 14, 15]. They provide flexible parameter selection as well as efficient encoding and decoding methods. As we will show in the next sections, an efficient ECC can greatly improve the performance of the fuzzy cryptographic algorithms.

2.1 Fuzzy cryptographic primitives for PS-based key exchange

We briefly describe all fuzzy cryptographic primitives evaluated in this paper and show their use for PS-based 128-bit key exchange. We distinguish two types: (i) based on Hamming distance (fuzzy commitment, code-offset and syndrome) and (ii) based on set difference (fuzzy vault, improved Juels–Sudan and Pinsketch).

We denote the transmitter and receiver that agree on a cryptographic key as TX and RX, and refer to the physiological values generated by TX and RX as ps and ps'. $\xleftarrow{Ext}$ denotes extraction of ps or ps' from raw PS measurements, and $\xleftarrow{R}$ denotes random number generation. $\xleftarrow{Shuffle}$ refers to randomly mixing elements in a set, while calculating the roots of a polynomial is denoted as $\xleftarrow{roots}$. We write $(0,1)^\ell$ for an ℓ-bit length string and $\{x, y\}^s$ for a set comprising s distinct elements. In all fuzzy cryptographic constructions described below, the first step is to extract ps and ps', which we will omit in the rest of this section.

In a *fuzzy commitment* [15], TX generates a random *key* and encodes it to form a *codeword* (Figure 1a). Subsequently, TX masks the *codeword* by XORing it with ps and then sends the resulting value (denoted by fc) to RX. Upon receiving fc, RX generates *codeword*$'$ by XORing fc with ps'. Only if the mismatch rate between *codeword* and *codeword*$'$ is less than the ECC's error tolerance limit, RX can successfully recover the key previously generated by TX.



(a) Fuzzy commitment (b) Fuzzy vault

Fig. 1: Fuzzy cryptographic primitives.

The *fuzzy vault* (Figure 1b) [14] is designed to "lock" a key using a set of features A. It can be unlocked only by using a set of features B that is sufficiently similar to A. Concretely, TX generates a key and embeds it in a univariate polynomial $p()$. Then, TX mixes and sends valid points (x, y), where x is in ps and $y = p(x)$, and invalid points (also known as 'chaff points') that do not lie on $p()$. For each received point, RX verifies whether x is in ps', and then performs polynomial reconstruction based on all the matched points. Only if the overlap between ps and ps' is sufficiently large, RX can successfully recover the key.

Both fuzzy commitment and fuzzy vault transport a key using two similar PS measurements. In contrast, *fuzzy extractors* [9] extract the key from the PS itself. Generally, the mismatches of PS measurements at TX and RX are corrected by sharing "helper data". Afterwards, both sides use the agreed PS to extract the cryptographic key with a strong random extractor (e.g., a secure hash function). We consider four fuzzy extractors in Hamming and set metrics. We omit the key extraction step below as it is the last step shared by all constructions.

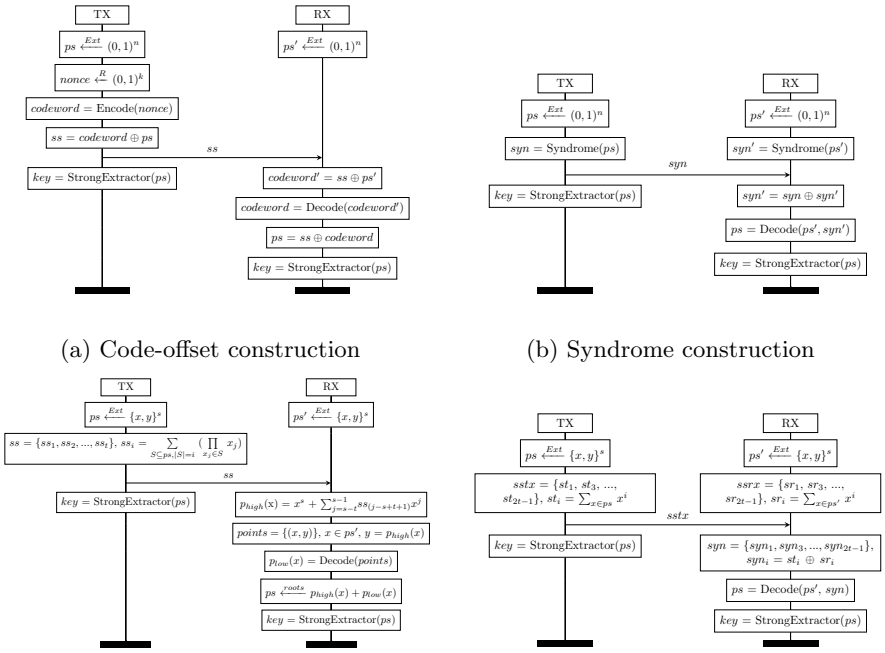
The *code-offset construction* (Figure 2a) is similar to the fuzzy commitment scheme, but here ps is the secret, while in fuzzy commitment, ps conceals the key. In particular, TX generates a random *nonce* and encodes it as *codeword* using the ECC. Then, TX sends $ss = codeword \oplus ps$. RX obtains the $codeword' = ss \oplus ps' = codeword \oplus ps \oplus ps'$ and can decode it to *codeword* if the mismatch rate is within bounds. Finally, RX recovers $ps = codeword \oplus ss$.

The *syndrome construction* (Figure 2b) is based on syndrome decoding of an ECC. Concretely, TX and RX regard ps and ps' as a codeword and calculate syndromes syn and syn', respectively. TX sends syn to RX, who calculates $syn \oplus syn'$. For mismatch vector $mis = ps \oplus ps'$, $syn \oplus syn'$ is the syndrome of mis, which decodes to mis if the mismatch rate is within bounds. Then, one recovers $ps = mis \oplus ps'$. Compared to code-offset construction, the syndrome is always shorter than the codeword, reducing the communication overhead.

In the *improved Juels–Sudan construction* (Figure 2c), TX uses the monic polynomial $p(x) = \prod_{w \in ps} (x - w)$ with roots as elements in ps and writes it as the sum $p_{high}() + p_{low}()$. TX calculates the coefficients of $p_{high}()$ and sends them

to RX. Then, RX generates points (x, y) where x is in ps' and $y = p_{high}(x)$. If $ps' \approx ps$, most points will also be on $p_{low}()$, so RX can reconstruct it and obtain ps by finding roots of $p()$. Compared with fuzzy vault, the communication overhead is much lower as only some coefficients have to be sent.

The *Pinsketch construction* (Figure 2d) is based on an ECC. For universe size $u = 2^m - 1$, a set *set* can be viewed as a vector $\{0, 1\}^u$, with 1 at position where $x \in set$ and 0 otherwise. In this way, ps and ps' are written as two such u-element vectors v, v' whose Hamming weight is the set size s. TX and RX calculate the syndromes $sstx$ and $ssrx$ of v and v'. Afterwords, TX sends $sstx$ to RX, while RX computes $syn = sstx \oplus ssrx$. If the mismatch rate is under the error tolerance of the ECC, the syndrome decoding result of syn is the symmetric difference between sets ps and ps', which helps correct the mismatches. Because v and v' are binary vectors, BCH codes are particularly suitable [9].



(a) Code-offset construction

(b) Syndrome construction

(c) Improved Juels–Sudan construction

(d) Pinsketch construction

Fig. 2: Fuzzy extractor constructions.

3 Design Security and Parameter Selection

To provide a systematic comparison and evaluation of fuzzy primitives for PS-based key exchange on resource-constrained embedded systems, we make several design decisions: Taking into account the limits on energy consumption and computational resources in a body-area network scenario and the fact that keys are

often short-lived, we limit ourselves to 128-bit keys. Furthermore, we only consider key exchange between two devices. We note that subsequent protocol steps, such as key confirmation step to ensure that TX and RX derive the same 128-bit key, are independent of the underlying fuzzy primitive and hence do not consider those steps. We also note that fuzzy primitives are only responsible for correcting the mismatches of the PS, *i.e.*, we do not consider errors on the wireless channel, and assume that the underlying wireless protocol includes appropriate error detection and correction measures.

Adversary model. We consider a strong adversary who knows all details about the used fuzzy primitives and has full access to the communication channel between TX and RX. The adversary can (i) perform *passive attacks* by eavesdropping on the communication and exploiting information leakage from it. For example, if ps and ps' in a fuzzy commitment are low-entropy, the adversary can statistically analyse their distribution and thus compromise the security [26]. Alternatively, correlation-based methods that leverage the correlation between communication data over *multiple* key exchange sessions can be used [16]. On the other hand, the adversary can also (ii) carry out *active attacks*, *i.e.*, act as Man-In-The-Middle (MITM) or replay old sessions. Finally, the adversary can also observe and exploit secret-dependent timing leakage e.g., the precise time between two protocol messages, both in passive and active attacks.

We assume that the measured PS cannot be modelled or predicted and cannot be remotely obtained. The latter implies that adversary can be in proximity to the user but cannot touch him directly or indirectly (because this would allow the adversary to measure the signal), nor being able to compromise a device worn by the user to measure the PS. In the research community, this *touch-to-access* access control model is widely accepted as it offers a reasonable trade-off between security and availability [23, 29]. Although the security may rely on user awareness to some extent, this model ensures high availability in emergency situations where fast establishment of a secure channel to the IMD is vital. For this reason, we leave physical side-channel and other attacks with direct access (such as fault injection) out of the adversary model, as in this case, the adversary can equally measure the PS directly for key recovery. We also do not consider Denial-of-Service (DoS) attacks such as jamming or battery depletion attacks.

Countermeasures against passive attacks. The underlying security of the fuzzy commitment against offline attacks depends on the entropy of the bit strings ps and ps' extracted from PS as these are used to conceal the key (by XOR) while being transported. The security of fuzzy vault relies on the fact that the adversary cannot distinguish between valid and chaff points, and hence is unable to reconstruct $p()$. For a fuzzy vault scheme with parameters g, b and d (cf. Figure 1b), the adversary would need an average of $\binom{g+b}{(g+d)/2}/\binom{g}{(g+d)/2}$ attempts to reveal $p()$ (assuming that Berlekamp-Welch decoding is used). Therefore, the number of chaff points needs to be sufficiently large.

For fuzzy extractors, the security depends on the entropy of the PS itself as the key is directly extracted from the PS. Due to the leakage of helper data, there

will be an amount of entropy loss on PS in each construction. For code-offset and syndrome constructions, the entropy loss is $(n-k) \cdot f$, where n, k come from the underlying (n, k, t) ECC, and f is the number of bits constituting each symbol. The entropy loss is $t \cdot \log_2 u$ for the improved Juels–Sudan construction, and is $t \cdot \log_2(u+1)$ for the Pinsketch construction, where u is the universe size and t is the maximum set difference between ps and ps'. Note that the above represent worst case entropy loss values [9]. Some of them were also proven to be overly pessimistic [8]. To ensure security, we regard the worst-case values as the actual entropy loss in this paper. Because we only focus on 128-bit key agreement, if the remaining entropy of PS (the agreed PS before input to the strong extractor) is ≥ 128 bit, the fuzzy extractor is considered secure.

For correlation-based attacks, note that in the case of PS-based key exchange, the PS has to vary over time and the exchanged key (generated randomly by the device or extracted from the PS) is short-lived and different in each session [19], unlike scenarios based on non-variable materials, e.g., biometrics or PUFs. This means that correlation attacks are prevented by the nature of the application.

Countermeasures against active and timing attacks. Due to the varying key, replay attacks are by design prevented. Other active attacks (such as guessing-based ones) require the adversary to break the fuzzy primitive "online" within a single protocol session [3, 17], otherwise, they at most result in failure of the key exchange and are detected by subsequent key confirmation. They can thus be prevented by generating a secret with substantially high entropy. Active attacks based on accurate measurement or modelling of the underlying PS [5] are outside our adversary model. Timing attacks can be generically prevented using constant-time implementations techniques, which we further discuss in Section 4.

Assumptions on physiological signal. The selection of the PS (e.g., IPI) and its quality as an entropy source, although an important issue, are out of the scope of this paper. However, we would like to stress that the quality of PS only affects the total measurement time, e.g., a lower quality of the entropy source means longer measurements. In order to generate the input for the fuzzy primitives, a set of pre-processing methods (e.g., quantisation and coding [25]) is applied to the raw PS measurements. However, this is out of the scope of this paper.

Assumptions on fuzzy primitive input. The inputs of fuzzy primitives (*i.e.*, ps and ps' in Section 2.1) are extracted from some PS which is measured by two devices simultaneously. There are several factors that affect the similarity of ps and ps', e.g., the type of PS, the measurement accuracy of the sensor, and the signal processing method. To evaluate and compare different fuzzy primitives, it is necessary to consider pre-defined mismatch rates (*i.e.*, percentage of different bits/set elements) between ps and ps', which reflect the characteristics of different kinds of scenarios. In this paper, we consider three mismatch thresholds of 2%, 5% and 10%. While the authors of [38, 39] reported that the mismatch rate for heart rate measurements is typically below 5%, we note that other PSs might have slightly higher mismatch thresholds. We also note that unlike BCH codes,

RS codes are multi-bit-symbol based. Thus, for RS code variants, the above thresholds indicate the percentage of different symbols rather than bits. Additionally, one should note that the average bit error rate (*i.e.*, the possibility that each bit differs for two bitstrings) may be more broadly used on the Hamming metric in other application scenarios, thus, we also provide this information for each variant in Table 2 (with a maximum tolerable failure rate of 10^{-6}). This maximum average bit rate that can be tolerated needs to be considered when selecting the most appropriate error correcting code.

Here, we assume that ps and ps' are ℓ-bit strings that are random and uniformly distributed for Hamming metric methods, or sets containing s distinct elements that are uniformly distributed in a universe U for set metric methods. This assumption is only made for fairly comparing different fuzzy primitives; both fuzzy commitment and fuzzy vault naturally require the input to be uniformly randomly distributed to ensure security[6]. Additionally, if the fuzzy primitive inputs are not uniformly distributed, it is hard to quantify the entropy level of the PS and establish a unified mismatch rate threshold. Under this assumption, the initial entropy of the PS is ℓ for Hamming metric fuzzy extractors, while for set metric fuzzy extractors, it is $log_2\binom{u}{s}$ with u the size of U.

Parameter selection. The mismatch rate between ps and ps' directly determines the error tolerance requirement of the fuzzy primitives. For Hamming metric methods, the error tolerance is the same as that of the underlying ECC (*i.e.*, t/n for an (n, k, t) code). For example, (50, 44, 1) and (20, 15, 1) BCH codes are suitable when the maximum mismatch rate is 2% and 5%, respectively. However, RS codes cannot provide exact 2%, 5% and 10% error tolerance because of their inherent structure. Therefore, we selected several RS constructions with error tolerance within 2% + the pre-defined mismatch rate thresholds. For example, the error tolerance of a (31, 29, 1) RS code is 3.23%. As mentioned in Section 2, all codes stay within the field $GF(2^8)$. For set metric methods with (u, s, t) structure, where u is the universe size, s the set size and t the maximum tolerable set difference between ps and ps', the error tolerance is $t/2s$.

For each fuzzy primitive, there can be multiple feasible parameter choices under the same mismatch rate, e.g., using different configurations of ECCs may achieve the same error tolerance. The difference between them is that the repetition count might be different: Assume the total repetition count is r and the number of secret bits distributed in each iteration is i, we need to ensure that $r \cdot i \geq 128$ to achieve 128-bit security. This way, RX concatenates the secret bits it receives in each iteration to form the 128-bit key. The security of subsequent/parallel execution based on linear codes has been proven in [8]. Note that i is the length of the key distributed per iteration for fuzzy commitment and fuzzy vault, but the remaining entropy of PS for fuzzy extractors. Consider a fuzzy commitment that is based on (50, 44, 1) and (200, 168, 4) BCH codes (both

[6] However, this requirement can be alleviated with the combination of a Password Authenticated Key Exchange (PAKE), as shown in [17]. Note that fuzzy extractors can still be securely used even if the inputs are not uniformly distributed.

handle mismatch $\leq 2\%$) as an example. In order to distribute a 128-bit key, the former variant needs to be executed three times ($3 \cdot 44 > 128$), while the latter only needs to be executed once. Although more iterations may be required, small parameter choices (e.g., an ECC with a small block size) almost always mean less computation and hence less energy consumption. Therefore, we test different variants under the same mismatch threshold. A variant with larger parameter choice is considered only if it reduces the number of required iterations. Besides, under each mismatch rate, the variants used by different fuzzy primitives in each metric are the same, thus help with the performance comparison.

Note that for Hamming metric methods, the number of feasible variants depends on the number of underlying ECCs that achieve the pre-defined mismatch thresholds. We give all feasible variants for Hamming metric in Table 2. For set metric methods, there are more possible (u, s, t) variants because (i) the universe size u can vary depending on how a PS is converted to a set and (ii) multiple s and t combinations can achieve the same error tolerance. In this paper, we use $u = 255$, which is the maximum universe size for the Pinsketch construction on $GF(2^8)$. We define three variants (255, 50, 2), (255, 20, 2) and (255, 10, 2) for 2%, 5%, and 10% mismatch rate thresholds. These variants are provided for reference only, and one could devise more appropriate variants for set metrics with specific mappings from PS to set. Finally, the fuzzy vault construction over $GF(2^8)$ is insecure, because the number of chaff points is $\leq 2^8$. However, for our performance evaluation in Section 5, we limit ourselves to $GF(2^8)$, and note that the system can be easily extended to larger fields (e.g., $GF(2^{16})$).

4 Implementation

We implemented, ran, and measured all algorithms on a TI MSP430FR5969 LaunchPad development board [34]. This board comprises a 16-bit microprocessor with 2 kB volatile SRAM and 64 kB permanent FRAM, which is representative for low-power body area network devices (including e.g., IMDs). We also alternatively used an MSP430FR5994 development board [35] with 8 kB SRAM for certain variants that require more resources, and indicate this in Table 2. For development, we used TI's Code Composer Studio as it provides integrated functionality for on-device energy consumption measurement.

Implementation of the strong extractor. Considering many embedded microcontrollers, including the MSP430 used in this paper, feature a hardware AES accelerator, we opted to use a block cipher-based hash function, with AES as the underlying cipher. We selected the Hirose construction [12] for the strong extractor in our implementation. Hirose is a double-block-length hash with Merkle–Damgård structure. We measured the average energy consumption of each invocation of the Hirose compression function on MSP430FR5969 to be 1.42 µJ. Depending on the availability of a fast hardware/software implementation, other hash functions such as SHA256 can be used instead of Hirose.

Software development and energy measurement. We implemented all algorithms in plain C and mainly relied on standard C libraries so that our implementation can be easily ported to other platforms. For random number generation and hardware-accelerated AES, we used TI's driver APIs. The tested average energy consumption of generating 16 bytes when using TI's random number generator API is 2.5 μJ. Certain components can be implemented in different ways. For example, there are a variety of algorithms for ECCs. We chose commonly used, efficient algorithms: for BCH and RS encoding, we used standard cyclic code encoding, and for decoding we used the Berlekamp decoding method [18]. For polynomial reconstruction, we used the Berlekamp-Welch algorithm.

We carried out the energy consumption measurement using TI `energyTrace` tool. This functionality allows to take accurate on-device energy measurement from the Code Composer Studio IDE. For each measurement, we averaged the energy consumption value over 100 executions of the respective algorithm. To test the error correction ability of the fuzzy primitives, we artificially added the maximum tolerable number of mismatches on the PS in the code, and then measured the corresponding energy consumption. We used TI `Ultra Low Power Advisor` tool to refactor our code and minimise energy consumption. Overall, we found that these optimisations reduced the energy consumption $\leq 10\%$.

Estimation of communication energy cost. The energy consumption of a protocol between multiple devices comprises two components: (i) the energy consumption of computations; and (ii) the energy consumption of wireless communication. In this paper, we only measure the energy consumption of computation, and model the cost of wireless communication based on the number of bits to be transmitted and received. In particular, in Section 5, we use the experimental results of [24] for a TelosB [7], a wireless sensor node based on a 16-bit MSP430 microcontroller and a CC2420 transceiver to illustrate the impact of communication overhead on overall energy consumption. Their results show that for 75 kbps data rate and -5 dBm transmit power, the average energy required to transmit one bit of effective data is 0.72 μJ, and the energy required to receive this bit is 0.81 μJ.

We acknowledge that a simplistic "energy-per-bit" model may be inadequate e.g., when using packet-based protocols such as Bluetooth Low Energy (BLE)[7], where the constant overheads due to the frame structure and other steps (e.g., wakeup and preparation) can be substantial. Therefore, we also provide the number of payload bits for each variant in Table 2, which can be fed into a more appropriate energy consumption model for a specific wireless protocol (e.g., informed by measurements as reported in [33]). We note that many widely used protocols support payloads large enough to accommodate all our variants in one packet (e.g., 246 bytes for BLE). This minimizes the impact of the frame structure, thus, when considering such protocols, the different implementations can be compared purely based on their computational energy cost.

Defenses against timing side channels. We implemented countermeasures against timing-based side-channel attacks on the best-performing variants (*i.e.*, fuzzy

[7] BLE is already being used in commercial IMDs e.g., Medtronic Azure pacemakers [1].

primitives with lowest total energy consumption under different mismatch thresholds, cf. Table 4). We found that non-constant execution time mainly arises in the ECC encoding/decoding processes through various conditional branches depending on a value being negative. To address this, we replaced all such conditional branches with Boolean operators, and used other constant-time implementation techniques, such as constant-time modulo reduction (based on Barrett reduction) and constant-time sorting.

		TX				RX			
		protected		unprotected		protected		unprotected	
fuzzy primitive	variant	t-value	energy(mJ)	t-value	energy(mJ)	t-value	energy(mJ)	t-value	energy(mJ)
Syndrome extractor	(31,29,1)RS	0.06	0.016	92.69	0.021	0.73	0.023	2039.85	0.029
Syndrome extractor	(31,27,2)RS	1.64	0.029	2220.64	0.037	1.67	0.043	352.48	0.053
Syndrome extractor	(63,49,7)RS	0.99	0.136	4927.79	0.198	0.80	0.237	438.42	0.308
Pinsketch extractor	(255,50,2)	1.01	0.089	242.24	0.046	0.72	0.168	733.98	0.127
Pinsketch extractor	(255,20,2)	0.77	0.052	468.67	0.044	0.91	0.208	428.64	0.201
Pinsketch extractor	(255,10,2)	0.20	0.066	298.66	0.067	6.63	0.377	236.67	0.380

Table 1: Effects of timing side-channel defenses on timing leakage (measured by Welch's t-test) and energy consumption.

We empirically verified the effects of implementing the above countermeasures, including the effect on timing leakage and energy consumption (communication overhead included). We used `dudect` [27] to evaluate the timing leakage of the TX and RX implementations running on the MSP430FR5969. The results are given in Table 1. The timing leakage of a program is evaluated by Welch's t-test in `dudect`. For each TX and RX implementation, the t-value in Table 1 was computed using 10,000 timing measurements. For a t-value ≤ 10, `dudect` regards the timing leakage as insignificant given the number of timing measurements.

It is evident that the baseline implementations exhibit strong timing leakage, while the protected variants significantly reduce the leakage below the constant-time threshold of the t-test in `dudect`. Besides, the energy consumption is not significantly increased for the protected variants. In fact, in some cases the energy consumption even decreases because of the use of Barrett reduction, which replaces the costly modulo operation otherwise implemented through division.

5 Performance evaluation

We implemented 22 variants of fuzzy primitives in total for the Hamming metric and three for the set metric. For each variant, we measured its computational energy consumption and estimated the communication cost at both TX and RX sides, and give the input size (extracted from the PS) required to achieve 128-bit security. Table 3 shows the main building blocks used by each fuzzy primitive.

In the following, we focus on the evaluation and comparison of selected variants. We include full, detailed results for all variants in Table 2. As mentioned, we base our estimation of communication costs on the values of $0.72\,\mu J$ per bit for TX and $0.81\,\mu J$ for RX [24], but also provide the number of exchanged payload bits for use with other models to estimate communication energy.

Table 2 shows the detailed measurement results for all considered fuzzy primitive instantiations in both the Hamming and set metric. We include the following characteristics of each variant: error tolerance, maximum average bit error rate, required number of iterations to achieve 128-bit security, computational energy cost at TX/RX (excluding communication cost), communication overhead (in bits transmitted/received), and required number of bits extracted from the PS. $*$ indicates implementation on MSP430FR5994 due to memory requirements.

					Fuzzy commitment			Code-offset construction			Syndrome construction		
Error tolerance	Max. average bit error rate	Variant	# iterations	PS data (bit)	TX (mJ)	RX (mJ)	Comm (bit)	TX (mJ)	RX (mJ)	Comm (bit)	TX (mJ)	RX (mJ)	Comm (bit)
2%	0.0016%	(50, 44, 1) BCH	3	150	0.022	0.079	150	0.030	0.091	150	0.052	0.084	36
2%	0.0146%	(100, 86, 2) BCH	2	200	0.037	0.201	200	0.045	0.215	200	0.127	0.207	56
2%	0.0854%	(200, 168, 4) BCH	1	200	0.082	0.389	200	0.087	0.404	200	0.244	0.394	64
3.23%	0.0037%	(31, 29, 1) RS	1	155	0.011	0.017	155	0.016	0.021	155	0.014	0.021	10
5%	0.0024%	(20, 15, 1) BCH	9	180	0.029	0.107	180	0.047	0.127	180	0.070	0.120	90
5%	0.0273%	(40, 28, 2) BCH	5	200	0.034	0.225	200	0.046	0.239	200	0.128	0.235	120
5%	0.0854%	(60, 42, 3) BCH	4	240	0.047	0.344	240	0.059	0.357	240	0.221	0.354	144
5%	0.1728%	(80, 52, 4) BCH	3	240	0.076	0.526	240	0.085	0.545	240	0.294	0.533	168
5%	0.2842%	(100, 65, 5) BCH	2	200	0.080	0.505	200	0.091	0.519	200	0.306	0.505	140
5%	0.9005%	(220, 136, 11) BCH	1	220	0.176	0.925*	220	0.179	0.976*	220	0.725	1.096*	176
6.67%	0.0038%	(15, 13, 1) RS	3	180	0.020	0.027	180	0.029	0.041	180	0.026	0.039	24
6.45%	0.0662%	(31, 27, 2) RS	1	155	0.019	0.034	155	0.023	0.040	155	0.023	0.037	20
10%	0.0031%	(10, 6, 1) BCH	22	220	0.060	0.141	220	0.099	0.180	220	0.103	0.180	176
10%	0.0407%	(20, 10, 2) BCH	13	260	0.046	0.307	260	0.073	0.331	260	0.175	0.331	260
10%	0.1429%	(30, 15, 3) BCH	9	270	0.044	0.399	270	0.065	0.420	270	0.259	0.428	270
10%	0.2904%	(40, 16, 4) BCH	8	320	0.067	0.732	320	0.085	0.745	320	0.394	0.735	384
10%	0.4822%	(50, 23, 5) BCH	6	300	0.074	0.786	300	0.091	0.797	300	0.452	0.785	360
10%	0.6855%	(60, 27, 6) BCH	5	300	0.077	0.878	300	0.092	0.896	300	0.548	0.885	360
10%	1.2713%	(90, 34, 9) BCH	4	360	0.131	1.713	360	0.145	1.730	360	0.965	1.724	504
10%	1.8153%	(120, 43, 12) BCH	3	360	0.156	2.102	360	0.168	2.110	360	1.276	2.085	504
10%	3.0090%	(210, 70, 21) BCH	2	420	0.288	3.340*	420	0.306	3.480*	420	2.637	4.025*	672
11.11%	2.4626%	(63, 49, 7) RS	1	378	0.103	0.247	378	0.107	0.252	378	0.138	0.240	84

		Fuzzy vault					Improved Juels–Sudan construction					Pinsketch construction				
Error tolerance	Variant	# iterations	TX (mJ)	RX (mJ)	PS data (bit)	Comm (bit)	# iterations	TX (mJ)	RX (mJ)	PS data (bit)	Comm (bit)	# iterations	TX (mJ)	RX (mJ)	PS data (bit)	Comm (bit)
2%	(255, 50, 2)	1	2.534	7.036*	400	4080	1	0.391	18.957*	400	16	1	0.034	0.114	400	16
5%	(255, 20, 2)	1	1.354	0.640	160	4080	2	0.132	2.980	320	32	2	0.021	0.175	320	32
10%	(255, 10, 2)	2	1.906	0.238	160	8160	4	0.073	2.320	320	64	4	0.021	0.328	320	64

Table 2: Evaluation of Hamming and set metric methods

TX side	RNG	ECC encoding	XOR	Hash	Syndrome gen.	Find poly. coeffs	Gen. points on $p()$	Gen. chaff points	Shuffle points
Fuzzy commitment	●	●	●						
Fuzzy vault	●						●	●	●
Code-offset extractor	●	●	●	●					
Syndrome extractor				●	●				
Improved JS extractor				●		●			
Pinsketch extractor				●	●				

RX side	ECC decoding	XOR	Reconstruct $p()$	Syndrome gen.	Hash	Gen. points on $p()$	Filter points	Find root on $p()$
Fuzzy commitment	●	●						
Fuzzy vault			●				●	
Code-offset extractor	●	●			●			
Syndrome extractor	●	●		●	●			
Improved JS extractor			●		●	●		●
Pinsketch extractor	●	●		●	●			

Table 3: Main building blocks of fuzzy primitives for TX and RX.

5.1 Hamming metric constructions

The minimum number of input bits derived from the PS is given as $n \cdot f \cdot r$, where n is the codeword length of the chosen ECC, f is the number of bits constituting a symbol, and r is the number of repetitions (*i.e.*, how many iterations of the primitive are required for 128-bit security). Depending on the specific variant, between 150 and 420 PS-derived bits are required (cf. Table 2). However, as we focus on the fuzzy primitive itself, rather than the conversion from PS to the algorithm input, we provide these values for reference only and to guide developer decisions in specific situations.

Computation costs. Figure 3 shows the energy cost of the Hamming metric fuzzy primitives. For each mismatch threshold, we show four variants of each fuzzy primitive and note that they are adequate to indicate the overall trend. At TX side, we observe that the fuzzy commitment consumes the least energy. The cost of the code-offset extractor is generally slightly higher than the fuzzy commitment under different mismatch rates. This result is in line with our expectations, because the code-offset extractor can be seen as a fuzzy commitment with additional invocation of a strong extractor. The syndrome fuzzy extractor involves the most energy-intensive computations. This is because syndrome generation for BCH and RS codes is an expensive operation that involves repeatedly evaluating $p(x)$ given x, which requires a number of iterative operations.

At RX side, we note that the energy consumption of the fuzzy commitment is also the smallest. The difference in computational energy consumption between the code-offset and syndrome extractors is often small because both extractors share several building blocks. Note that, even if the syndrome extractor has an extra "syndrome generation" block compared to the code-offset extractor (cf. Table 3), the actual execution of these constructions is equivalent.

Regarding ECC choice, the RS code performs substantially better than BCH for mismatch rates below 5%. However, for 10% mismatch, the chosen RS code is worse than the best BCH variant. However, note that e.g., the (63, 49, 7) RS instance can accommodate up to 343-bit distribution per iteration for the fuzzy commitment, while we only require 128 bits, which is not optimal if only considering computational energy consumption.

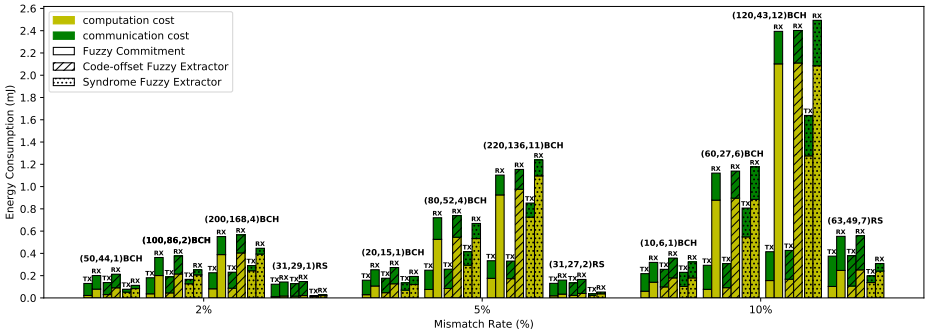


Fig. 3: Energy consumption of Hamming metric primitives.

Combined computation and communication cost. When we also take the estimated communication costs into account, the syndrome fuzzy extractor outperforms the other two variants most of the time. For both fuzzy commitment and code-offset construction, the communication overhead is determined by the codeword length n of the chosen ECC and the number of required repetitions. For example, consider the (50, 44, 1) BCH variant in Table 2. In this case, the communication overhead is $50 \cdot 3$ bits, because the variant needs to be executed three times to establish a 128-bit key. In contrast, the communication overhead for the syndrome extractor depends on the syndrome length and the number of repetitions. The length of the syndrome is $2 \cdot t \cdot m$ for BCH and RS codes, where m comes from $GF(2^m)$ underlying the ECC [18]. An obvious advantage is that the syndrome is always shorter than the codeword. Considering the previous example, TX would only need to transmit $12 \cdot 3$ bits for the syndrome extractor. Overall, the variants with lowest combined computation and communication cost under each mismatch rate are shown in Table 4. The syndrome extractor has variants with the lowest total energy consumption in all conditions.

mismatch rate	fuzzy primitive	variant	total energy at TX (mJ)	total energy at RX (mJ)
2%	Syndrome extractor	(31,29,1) RS	0.021	0.029
5%	Syndrome extractor	(31,27,2) RS	0.037	0.053
10%	Syndrome extractor	(63,49,7) RS	0.198	0.308
2%	Pinsketch extractor	(255,50,2)	0.046	0.127
5%	Pinsketch extractor	(255,20,2)	0.044	0.201
10%	Pinsketch extractor	(255,10,2)	0.067	0.380

Table 4: Fuzzy primitives with lowest total energy cost on the Hamming metric (Syndrome extractor) and set metric (Pinsketch extractor).

5.2 Set metric constructions

The required number of derived bits for set metric methods is $s \cdot f \cdot r$, where s is the set size, f is the number of bits constituting a set element, and r is the number of repetitions. Set metric constructions require input sizes from 160 to 400 bits (cf. Table 2), which is similar to the Hamming metric variants.

Computation costs. Figure 4 shows the energy consumption of all considered set metric methods (cf. Table 2 for the underlying data). At TX, we observe that

the fuzzy vault consumes substantially more energy than the other two methods. This is likely because TX of the fuzzy vault has to generate and shuffle a large amount of points (mostly chaff points). The TX energy cost of the Pinsketch fuzzy extractor is slightly below improved Juels–Sudan. According to Table 3, the difference in energy consumption is due to the difference between syndrome generation (note that this is not the same as the standard syndrome calculation of BCH and RS code) and the polynomial coefficient finding.

On the RX side, we find that the Pinsketch construction has the lowest energy consumption for mismatch rates below 5%, and has slightly higher consumption than fuzzy vault for 10% mismatch. The improved Juels-Sudan fuzzy extractor is always the most expensive construction under all mismatch rates, likely due to the required polynomial root finding process. We further observe that the energy consumption of this method and the fuzzy vault decrease significantly when the mismatch rate threshold increases. This is expected because both methods rely on the same complex polynomial reconstruction for mismatch correction. This involves operations on $s \times s$ matrices (s is the set size). Hence, polynomial reconstruction is efficient for small sets (*i.e.*, under higher mismatch threshold), but as the set size increases, the computational complexity increases quadratically.

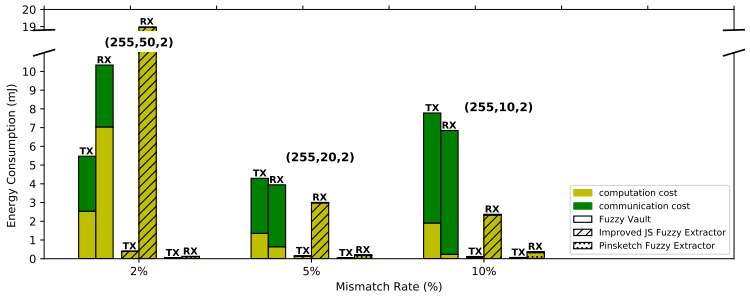


Fig. 4: Energy consumption of set metric primitives.

Combined computation and communication cost. The number of transmitted bits for the fuzzy vault is $np \cdot lp \cdot r$, where np is the total number of points (valid and chaff points), lp is the length of each point, and r is the repetition count. For our universe size of 255, $np = 255$ and $lp = 16$ bits (each coordinate is one byte). In contrast, the transmission size for improved Juels-Sudan and Pinsketch fuzzy extractors is $t \cdot f \cdot r$, where t is the maximum tolerable set difference between sets and f is the number of bits constituting a set element (in our constructions $f = 8$). Hence, the communication cost of the fuzzy vault is much higher compared to improved Juels-Sudan and Pinsketch. The overall best set metric variants for each mismatch threshold are shown in Table 4. The Pinsketch fuzzy extractor performs best in terms of combined computation and communication cost in all cases. The improved Juels-Sudan has extremely high computation cost in RX, while the fuzzy vault incurs substantial communication overhead. Considering that secure implementation of fuzzy vault requires operations over $GF(2^{16})$ and transmits more points, it is likely that costs would further grow in practice.

5.3 Common observations and comparison with Curve25519

We observed certain common tendencies for all fuzzy primitives: for each variant, the computation energy consumption for RX is generally higher than for TX. Conversely, the communication cost is roughly the same for TX and RX. This observation is relevant when assigning TX/RX roles in more complex protocols; e.g., a low-power IMD can act as TX if the goal is to minimise energy consumption. In addition, the energy consumption of variants with larger parameter (e.g., larger BCH code) shows an increase both for TX and RX, even though the number of required repetitions decreases. However, we note that such variants can always handle higher average bit error rate (cf. Table 2).

We compared the energy cost of our best-performing variants with Curve25519, one of the most efficient elliptic curve-based key exchange schemes for embedded systems. As reported in [11], one full execution of Curve25519 on MSP430FR5969 costs about 0.012 mJ (0.404 mJ if communication energy is estimated as in this paper). Thus, our methods are comparable in terms of total energy consumption. Moreover, fuzzy primitives provide security guarantees beyond a public-key scheme such as Curve25519: they can defend against MITM attacks (without certificate infrastructure) and guarantee that RX and TX are in physical proximity.

6 Conclusion

In this paper, we systematically and fairly evaluate the performance of fuzzy cryptographic primitives for PS-based key exchange under controlled conditions on a resource-constrained MSP430 microcontroller. We show how different fuzzy primitives can be securely applied to derive a 128-bit key from joint measurements of a PS, and provide implementations of each of these primitives in multiple variants. To our knowledge, we are the first to compare the computation and communication energy consumption of different fuzzy primitives for a variety of parameter choices. Among all considered fuzzy primitives, we find that Syndrome and Pinsketch fuzzy extractors overall offer the lowest energy consumption in Hamming and set metric spaces.

This indicates that fuzzy commitment and fuzzy vault used in previous PS-based key exchange solutions [3, 6, 13, 36, 37] are not optimal on constrained devices. Instead, Syndrome/Pinsketch fuzzy extractors may be preferable, with the added advantage that they neither require random number generation, which can be costly on embedded systems, nor uniformly randomly distributed inputs derived from a PS. These constructions consume between 0.021 mJ and 0.198 mJ for TX and between 0.029 mJ and 0.380 mJ for RX, including computational and communication energy. This demonstrates that PS-based key exchange methods using fuzzy primitives are feasible for a resource-constrained device, even if keys are relatively frequently exchanged. We also observe that ECCs with smaller parameter choices in fuzzy primitives have generally better performance, even if more repetitions are required. However, this might come at the cost of having more strict constraints on the maximum average bit error rate of a PS. Our work

serves as a reference when applying fuzzy primitives for body-area networks and medical devices, and for other use cases such as biometrics or PUFs.

Acknowledgements. This work is funded in part by the European Union's Horizon 2020 Research and innovation program under grant agreement No. 826284 (ProTego), the FWO-SBO project SPITE, and by the Engineering and Physical Sciences Research Council (EPSRC) under grant EP/R012598/1. Mo Zhang is funded by the Priestley PhD Scholarship programme.

References

1. Medtronic azure pacing system, https://europe.medtronic.com/xd-en/healthcare-professionals/products/cardiac-rhythm/pacemakers/azure.html
2. Abidin, A., Argones Rúa, E., Peeters, R.: Uncoupling Biometrics from Templates for Secure and Privacy-Preserving Authentication. In: ACM SACMAT (2017)
3. Al Reshan, M., Liu, H., Hu, C., Yu, J.: Mbpska: Multi-biometric and physiological signal-based key agreement for body area networks. IEEE Access **7** (2019)
4. Billeb, S., Rathgeb, C., Reininger, H., Kasper, K., Busch, C.: Biometric template protection for speaker recognition based on universal background models. IET Biometrics **4**(2), 116–126 (2015)
5. Calleja, A., Peris-Lopez, P., Tapiador, J.E.: Electrical Heart Signals can be Monitored from the Moon: Security Implications for IPI-Based Protocols. In: WISTP. pp. 36–51 (2015)
6. Cherukuri, S., Venkatasubramanian, K.K., Gupta, S.K.S.: Biosec: a biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In: ICPP. pp. 432–439 (2003)
7. Crossbow Technology Inc: TelosB datasheet, rev. B
8. Delvaux, J., Gu, D., Schellekens, D., Verbauwhede, I.: Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. IEEE TCAD **34**(6) (2015)
9. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. SIAM Journal on Computing **38**(1), 97–139 (2008)
10. Halperin, D., Heydt-Benjamin, T.S., Fu, K., Kohno, T., Maisel, W.H.: Security and Privacy for Implantable Medical Devices. IEEE Pervasive Computing, Special Issue on Implantable Electronics **7**, 30–39 (2008)
11. Hinterwälder, G., Moradi, A., Hutter, M., Schwabe, P., Paar, C.: Full-size high-security ECC implementation on MSP430 microcontrollers. In: LATINCRYPT. pp. 31–47. Springer (2014)
12. Hirose, S.: Some Plausible Constructions of Double-Block-Length Hash Functions. In: FSE. pp. 210–225 (2006)
13. Hu, C., Cheng, X., Zhang, F., Wu, D., Liao, X., Chen, D.: OPFKA: Secure and efficient Ordered-Physiological-Feature-based key agreement for wireless Body Area Networks. In: INFOCOM (2013)
14. Juels, A., Sudan, M.: A Fuzzy Vault Scheme. Designs, Codes and Cryptography **38**(2), 237–257 (2006)
15. Juels, A., Wattenberg, M.: A Fuzzy Commitment Scheme. In: ACM CCS (1999)
16. Kholmatov, A., Yanikoglu, B.: Realization of correlation attack against the fuzzy vault scheme. In: Security, forensics, steganography, and watermarking of multimedia contents X. vol. 6819, p. 68190O. SPIE (2008)

17. Li, X., Zeng, Q., Luo, L., Luo, T.: T2Pair: Secure and Usable Pairing for Heterogeneous IoT Devices. In: ACM CCS. pp. 309–323 (2020)
18. Lin, S., Costello, D.J.: Error control coding, vol. 2. Prentice hall (2001)
19. Marin, E., Argones Rúa, E., Singelée, D., Preneel, B.: On the Difficulty of Using Patient's Physiological Signals in Cryptographic Protocols. In: ACM SACMAT. pp. 113–122 (2019)
20. Marin, E., Mustafa, M.A., Singelée, D., Preneel, B.: A Privacy-Preserving Remote Healthcare System Offering End-to-End Security. In: AdHoc-Now (2016)
21. Marin, E., Singelée, D., Garcia, F.D., Chothia, T., Willems, R., Preneel, B.: On the (in)Security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them. In: ACSAC. pp. 226–236 (2016)
22. Marin, E., Singelée, D., Yang, B., Verbauwhede, I., Preneel, B.: On the Feasibility of Cryptography for a Wireless Insulin Pump System. In: CODASPY (2016)
23. Marin, E., Singelée, D., Yang, B., Volski, V., Vandenbosch, G.A.E., Nuttin, B., Preneel, B.: Securing wireless neurostimulators. In: CODASPY. pp. 287–298 (2018)
24. de Meulenaer, G., Gosset, F., Standaert, F., Pereira, O.: On the Energy Cost of Communication and Cryptography in Wireless Sensors Networks. In: IEEE WiMob. pp. 580–585 (2008)
25. Ortiz Martin, L., Picazo-Sanchez, P., Peris-Lopez, P., Tapiador, J.: Heartbeats do not make good pseudo-random number generators: An analysis of the randomness of inter-pulse intervals. Entropy **20**, 94 (2018)
26. Rathgeb, C., Uhl, A.: Statistical attack against fuzzy commitment scheme. IET biometrics **1**(2), 94–104 (2012)
27. Reparaz, O., Balasch, J., Verbauwhede, I.: Dude, is my code constant time? In: DATE. pp. 1697–1702. IEEE (2017)
28. Reverberi, L., Oswald, D.: Breaking (and Fixing) a Widely Used Continuous Glucose Monitoring System. In: USENIX WOOT (2017)
29. Rostami, M., Juels, A., Koushanfar, F.: Heart-to-Heart (H2H): Authentication for Implanted Medical Devices. In: ACM CCS. pp. 1099–1112 (2013)
30. Seepers, R.M., Strydis, C., Peris-Lopez, P., Sourdis, I., Zeeuw, C.I.D.: Peak misdetection in heart-beat-based security: Characterization and tolerance. In: EMBC. pp. 5401–5405 (2014)
31. Seepers, R.M., Wang, W., de Haan, G., Sourdis, I., Strydis, C.: Attacks on Heartbeat-Based Security Using Remote Photoplethysmography. IEEE J-BHI **22**(3), 714–721 (2018)
32. Singelée, D., Seys, S., Batina, L., Verbauwhede, I.: The energy budget for wireless security: Extended version. IACR Cryptol. ePrint Arch. **2015**, 1029 (2015)
33. TI: AN092: Measuring Bluetooth Low Energy Power Consumption (2012)
34. TI: MSP430FR596x, MSP430FR594x Mixed-Signal Microcontrollers datasheet (2012), https://www.ti.com/lit/gpn/msp430fr5969, rev. G
35. TI: MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers datasheet (2016), https://www.ti.com/lit/gpn/msp430fr5994, rev. C
36. Venkatasubramanian, K.K., Banerjee, A., Gupta, S.: Plethysmogram-based secure inter-sensor communication in body area networks. In: IEEE MILCOM (2008)
37. Venkatasubramanian, K.K., Banerjee, A., Gupta, S.K.S.: PSKA: Usable and Secure Key Agreement Scheme for Body Area Networks. IEEE T-ITB **14**(1), 60–68 (2010)
38. Venkatasubramanian, K.K., Gupta, S.K.S.: Physiological value-based efficient usable security solutions for body sensor networks. ACM TOSN **6**(4) (2010)
39. Xu, F., Qin, Z., Tan, C.C., Wang, B., Li, Q.: IMDGuard: Securing implantable medical devices with the external wearable guardian. In: IEEE INFOCOM (2011)