On the Energy Cost of Channel Based Key Agreement

Christopher Huth
Robert Bosch GmbH
Research and Advance Engineering
Stuttgart, Germany
christopher.huth@de.bosch.com

René Guillaume
Robert Bosch GmbH
Research and Advance Engineering
Stuttgart, Germany
rene.guillaume@de.bosch.com

Paul Duplys
Robert Bosch GmbH
Research and Advance Engineering
Stuttgart, Germany
paul.duplys@de.bosch.com

Kumaragurubaran Velmurugan
Robert Bosch GmbH
Research and Advance Engineering
Stuttgart, Germany
gurougem@gmail.com

Tim Güneysu
University of Bremen & DFKI,
Bremen, Germany
tim.gueneysu@uni-bremen.de

ABSTRACT

Besides security, energy consumption is a major concern for devices in the Internet of Things (IoT). We compare the energy consumption of two key agreement schemes – Channel-Based Key Agreement (CBKA) and Elliptic Curve Diffie-Hellman (ECDH) – in the IoT setting, using Wi-Fi as wireless communication interface. While ECDH is a well-studied protocol, CBKA has received attention only in recent years. Several publications proposed CBKA as a low-energy alternative to ECDH, but they did not address the energy cost of communication. For a fair comparison, we implemented the schemes on a 32-bit ARM Cortex M3-based IoT platform and measured the respective energy consumption for computation and communication. Our results show that the limiting factor for CBKA over Wi-Fi is the energy cost of communication, in particular the cost of acquiring the Received Signal Strength Indicator (RSSI) values. Even in an optimal scenario, CBKA must not measure more than ca. 300 RSSI values to be more energy efficient than ECDH. This is at most 1/5 of RSSI values required by CBKA implementations reported in the literature. As an optimization, we present a refined CBKA protocol which can save up to 25 % of the energy compared to existing protocols by exploiting inherent data exchanges for entropy extraction.

CCS Concepts

•**Hardware** → **Power estimation and optimization;** •**Security and privacy** → **Mobile and wireless security;** *Hardware security implementation; Security protocols;* Embedded systems security;

TrustED'16, October 28 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2995289.2995291

Keywords

Physical Layer Security; Energy Consumption; Reciprocal Radio Channels

1. INTRODUCTION

In an era of individualization and automation, sensors and actuators, and particularly of embedded devices with wireless connectivity, the merit of data is steadily increasing. Besides extracting and processing relevant information, protecting sensitive data from abuse is at least of equivalent significance.

For this reason, in recent years, many researchers strove toward innovative solutions for secret key agreement – especially suitable for scenarios found in the IoT, e.g. [12]. These typically comprise a network of a huge number of embedded devices, i.e. nodes that are highly limited in computational resources. Exemplary considering sensors and actuators, these battery-powered *things* often come without any kind of graphical or other user interface which allows the user to enter, e.g., passwords. Because of such constraints, it may not be feasible to apply classical approaches for key establishment (e.g. Diffie-Hellman Key Exchange (DHKE)). These often require complex arithmetical computation, hence put a high strain onto the device's battery power, and often depend on individual inputs by the user.

Previously summarized constraints introduced by embedded devices and the procession of Physical Layer Security led to the celebrity of Channel-Based Key Agreement, which has been subject to extensive academic as well as industrial research in recent years [22, 17, 14, 1, 8]. Many related papers dealt with individual aspects of CBKA, particularly addressing optimized signal processing and key generation rate (e.g. [19, 7]). However, only few referred to the actual motivation of CBKA, that is decreased energy consumption and feasibility for embedded devices (cf. [28]). A direct comparison to DHKE under practically relevant conditions remained unsettled.

1.1 Contribution

The underlying motivation for using ECDH and CBKA is equivalent: both result in a shared secret key. However, as the way of deriving this key is rather different, also the energy consumption can be expected to differ accordingly. Referring to the previously identified lack of practically relevant comparison dedicated to the energy consumption of ECDH and CBKA, this paper aims to answer the question of how these two key agreement schemes can be compared in a fair manner. While ECDH is a widely accepted scheme with well optimized implementations, CBKA is a promising alternative with a lot of potential still to be analyzed in modern security research. Though a major advantage of CBKA is its low computation complexity, little effort has been put into analyzing CBKA's energy consumption on typical IoT devices [5]. This particularly includes analysis with regard to CBKA (e.g. [27]).

We focus on resource-limited embedded devices to evaluate our findings on energy consumption of ECDH for various curve domain parameters versus CBKA for various quantization and reconciliation schemes. Our system utilizes state-of-the-art protocols. To summarize, our contributions are:

- We implement and evaluate various ECDH and CBKA schemes to provide energy consumption details for all protocols' substeps and perform extensive energy measurements.
- We present a model to estimate the energy for these schemes alike. The model allows us to estimate a lower bound on the energy consumption for CBKA in order to directly compare it to ECDH.
- We present a modification of the CBKA protocol that enables us to reduce the overall energy consumption by up to 25 %.

1.2 Related work

Measuring the energy consumption for generic software running on a given embedded hardware has already been carried out in various research, e.g. [26, 24]. Some research has been done to analyze if ECDH can run on very low power devices, even on 8-bit microcontrollers [15, 9, 23, 16]. In [5], de Meulenaer *et al.* analyzed the energy cost of two key agreement algorithms, Kerberos and Diffie-Hellmann key exchange, on a wireless sensor node involving ZigBee wireless interface.

Research on the impact of the wireless communication interface on any given embedded device is studied to a greater extend in [20]. However, there is no extensive research on the comparison of energy consumption between shared secret key agreement schemes. Recently, Zenger *et al.* compared the energy consumption of ECDH and CBKA [28]. Nevertheless, one of the inherent limitation of this research was that the channel measurements are assumed to be available for free in terms of energy. But in reality the energy cost for data transmission over a wireless channel can outweigh all other required energy, depending upon how many messages need to be exchanged for entropy extraction. Moreover, only a BCH-based reconciliation scheme was considered.

1.3 Outline

First, preliminaries are introduced in Section 2 for CBKA. In Section 3, we describe how we conducted our measurements for the computational and radio energy. Next, in Section 4 we provide a model to estimate the energy consumption introduced by ECDH and CBKA, respectively. We present in-depth analysis of needed substeps for key derivation. We measure the energy cost for a key in a realistic scenario and provide a lower bound for CBKA's energy consumption, based on these measurements. The results are compared in Section 5. Finally, we propose an improved CBKA scheme to decrease the energy consumption in Section 6. We discuss our results in Section 7 and conclude this article in Section 8.

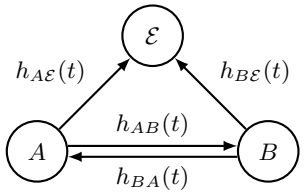

Figure 1: Two legitimate nodes A and B communicate via a wireless channel $h_{BA} \approx h_{AB}$ and exploit its reciprocity to agree on a shared secret key. An eavesdropper $\mathcal{E}$ experiences the channels $h_{A\mathcal{E}}$ and $h_{B\mathcal{E}}$. These can be assumed to be sufficiently uncorrelated to h_{BA} for certain distance between A and $\mathcal{E}$ as well as between B and $\mathcal{E}$, respectively.

2. PRELIMINARIES

The fundamental idea of CBKA is to exploit reciprocal properties of a fading radio channel to agree on a shared secret. We consider a system model according to Figure 1. Let us assume two legitimate nodes A and B communicating via a wireless broadcast link and an attacker $\mathcal{E}$ eavesdropping on the communication between A and B. The legitimate nodes A and B are considered to be successfully authenticated by any feasible means. The wireless channel from A to B is denoted h_{AB}, while h_{BA} indicates the backward channel direction from B to A. At one time instance t_0, the communication link can be assumed to be reciprocal. In practice it typically holds that $h_{BA}(t_0) = h_{AB}(t_0)$. Due to temporal correlation with coherence time τ_c, reciprocity approximately also holds with $h_{BA}(t_0) \approx h_{AB}(t_0 + \tau_c)$. This notation equivalently holds for other radio channels such as $h_{A\mathcal{E}}$ between nodes A and $\mathcal{E}$ and $h_{B\mathcal{E}}$ between nodes B and $\mathcal{E}$. Due to the reason of spatial coherence, the channels $h_{A\mathcal{E}}$ and $h_{B\mathcal{E}}$ may correlate to h_{BA}. This typically depends on the distances between A and $\mathcal{E}$ as well as between B and $\mathcal{E}$.

Our assumptions on attacker $\mathcal{E}$ are as follows. We consider node $\mathcal{E}$ to be a passive attacker, hence it eavesdrops on the exchanged data of the legitimate nodes A and B. The attacker tries to read as much as possible of the broadcasted information, but does not manipulate exchanged packets nor does it run denial-of-service attacks. Since authentication schemes are successfully applied, $\mathcal{E}$ is neither capable of running Man-in-the-Middle attacks. However, $\mathcal{E}$ may try to exploit the correlation between its channel observations $h_{A\mathcal{E}}$ and $h_{B\mathcal{E}}$ and the channel h_{BA} between A and B to derive parts of the symmetric key.

The process of generating keys with CBKA can be characterized by different building blocks, namely channel mea-

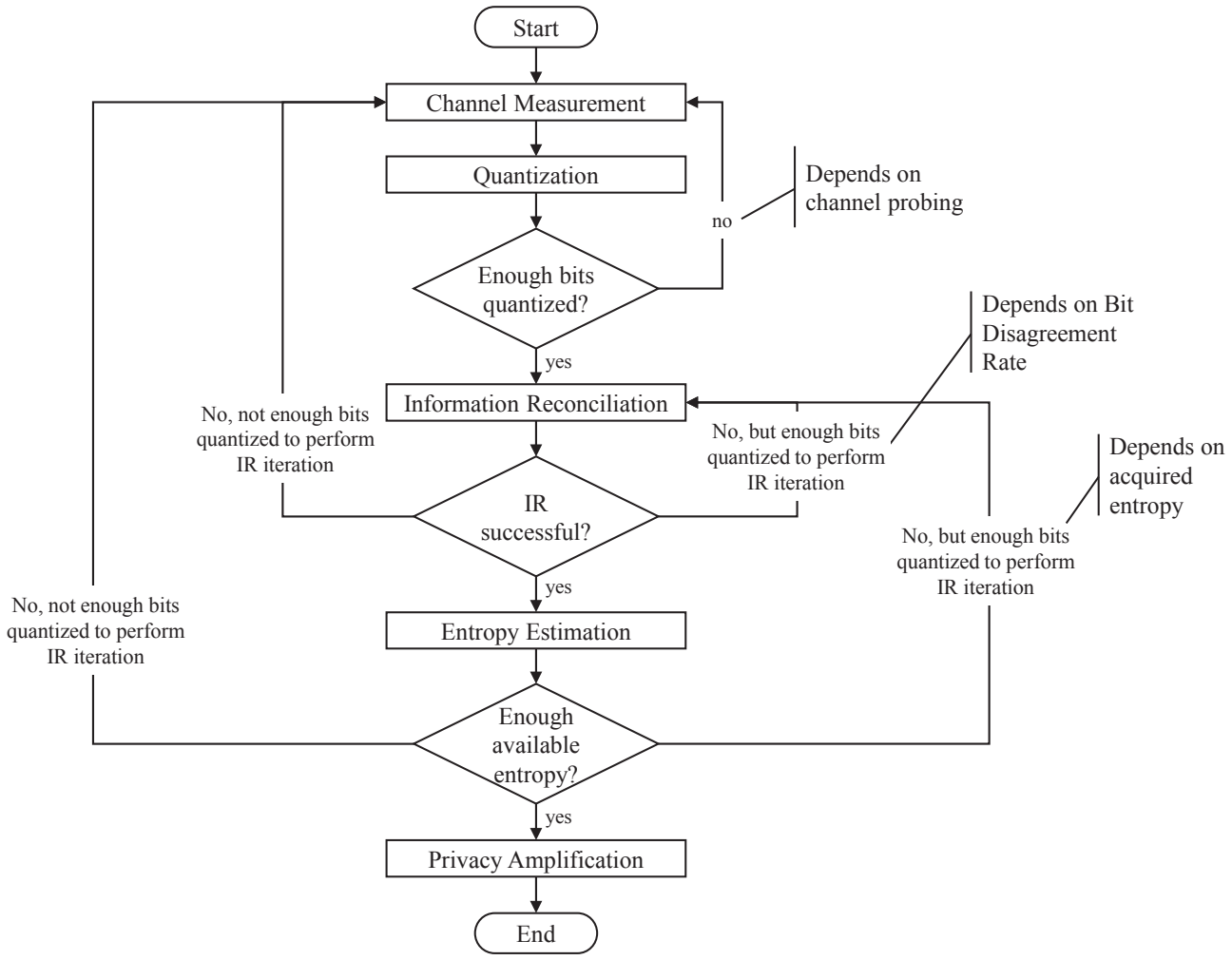

Figure 2: CBKA iteration flowchart.

surement, quantization, information reconciliation, entropy estimation and privacy amplification. These modules are combined in an iterative structure, depicted in Figure 2. Obtaining reciprocal channel properties in the *channel measurement* block represents the actual source of common randomness. In *quantization*, the measurement sequences are quantized to binary representations. Due to imperfect reciprocity, yet the sequences of A and B do not perfectly match. In order to address these bit disagreements, appropriate alignment protocols are applied in *information reconciliation*. For the purpose of monitoring the predictability of the extracted secret, *entropy estimation* is performed. Finally, in *privacy amplification*, e.g. cryptographic hash functions produce a secret key from the obtained random strings.

By applying different performance metrics after each of these modules, these blocks can be run iteratively in a loop. For instance, if the reciprocity of the measured channel characteristics is poor, the Bit Disagreement Ratio (BDR) between A and B is too high to allow efficient processing, as the reconciliation may not be able to correct all mismatches. Similarly, if the collected entropy is too low, the generated key may not fit the required security level. The loop back allows parts of the algorithm to be repeated accordingly.

3. MEASUREMENT SETUP

We used the EFM32GG-STK3700 Giant Gecko Starter Kit from Silicon Labs for measuring the computational energy consumption. It features a 32-bit ARM Cortex-M3 microcontroller and an on-board advanced energy monitoring system, allowing one to perform real-time voltage and current profiling of applications without using any external tools [21].

Since there are no standard libraries for CBKA available, we implemented several quantization schemes and two information reconciliation schemes as described in Section 4.2.

Our mobile scenario for the measurement of RSSI values consists of a stationary Bob and a moving hand-held Alice. We tried to cover many environmental situations, e.g., line of sight, line of sight blocked by different objects, communication through walls or distances ranging from thirty centimeters to twenty meters away. This way we introduced a dynamic behavior in the channel, so that we could later choose the best for key generation.

For measuring the energy consumption of the Wi-Fi radio module we used a high resolution oscilloscope. Our module is an AR4100 Wireless LAN IEEE 802.11b/g/n SIP. A $1\,\Omega$ resistor was connected in series to the power supply and

the target device. Everytime the Wi-Fi radio module transmits or receives data, measurements are triggering through increased drawn current. Depending on the instantaneous current drawn by the target device, a potential drop was observable across the resistor. This potential drop was measured by the oscilloscope using an active differential probe.

We found that the energy consumption of the Wi-Fi module is defined by the time it is turned on, which is controlled by the firmware. During this time the module transmits, listens and receives. We measured a power-on time of 3 ms and roughly the same energy for transmitting and receiving one packet via Wi-Fi:

- Tx: 20.767 μJ per packet
- Rx: 21.005 μJ per packet

This is in line with related work of De Meulenaer *et al.* [5], who reports 0.60 μJ for transmitting and 0.67 μJ for receiving one *single* bit on a MICAz, and 0.72 μJ for transmitting and 0.81 μJ for receiving one single bit on a TelosB.

4. ENERGY MODEL AND RESULTS

In this section, we present details on the energy consumption for key derivation. Our focus is on determining the cost of individual substeps for CBKA and ECDH alike.

4.1 Elliptic Curve Diffie-Hellman

Any ECDH implementation is based on the mathematical operations performed on an elliptic curve, such as addition, subtraction, multiplication, squaring and division. The algorithms can be realized in different ways, e.g., solely addition is used to implement the multiplication. This causes variations in the energy consumption of the individual arithmetic operations.

Still, for a single shared secret key generation, ECDH generates two random numbers irrespective of the chosen curve domain parameters. For random number generation we used the on-chip AES module in counter mode with a systick poll counter as a seed. This passes the tests in the NIST Statistical Test Suite [18]. We measured the following energy:

- 1.28 μJ per 128-bit random number

A general equation for energy consumption of ECDH algorithms is simply the sum of all subsequent operations' costs, the energy needed for Wi-Fi communication and some overhead, as in Equation (1). Equation (2) represents the energy consumption for the NIST curves and Equation (3) refers to curve25519 [3].

$$P_{\text{ECDH\_total}} = \left(\sum_{i \in API} N_i \cdot P_i \right) + P_{\text{RNG}} + P_{\text{Wi-Fi}} + P_{\text{Overhead}}, \quad (1)$$

where i represents an element from the set API of possible API calls, i.e., {add,sub,...}, N_i represents the number of times each API i is executed and P_i represents the energy consumption per API i call. P_{RNG} is the energy used for one random number generation and depends upon the used generator. The energy per API call P_i depends upon the generated random number and the chosen curve domain parameters, as they determine the number of API calls N_i. We measured all implementations for 2000 key generations and mapped the function calls on their corresponding average energy consumption. Upon analysis we found that the above assumption regarding the arithmetic operations holds and that the energy contribution of the Wi-Fi data exchange is negligible compared to the overall energy consumption.

In micro-ECC[1], library functions are implemented calling each other. For instance, the square function is implemented through the calls to APIs vli_mmod_fast, uECC_vli_square and uECC_vli_modSquare_fast. For our model, we explicitly state APIs that have a major contribution to the total energy, for the sake of simplicity. APIs that have a controlling nature, rather than for actual calculation, are added to Overhead. The API uECC_vli_add calls an addition operation, the APIs uECC_vli_clear, uECC_vli_set and uECC_vli_cmp_unsafe are bit operations and vli_mmod_fast is a modular multiplication. Furthermore, we split the energy for Wi-Fi into parts for transmitting P_{Tx}, receiving P_{Rx} and listening P_{Listen}. Equation (2) represents the total energy consumption based on the number of API calls for NIST curves.

$$\begin{aligned} P_{\text{ECDH\_NIST\_total}} = {} & (N_{\text{uECC\_vli\_square}} \cdot P_{\text{uECC\_vli\_square}}) \\ & + (N_{\text{vli\_mmod\_fast}} \cdot P_{\text{vli\_mmod\_fast}}) \\ & + (N_{\text{uECC\_vli\_modSquare\_fast}} \cdot P_{\text{uECC\_vli\_modSquare\_fast}}) \\ & + (N_{\text{uECC\_vli\_mult}} \cdot P_{\text{uECC\_vli\_mult}}) \\ & + (N_{\text{uECC\_vli\_rshift1}} \cdot P_{\text{uECC\_vli\_rshift1}}) \\ & + (N_{\text{uECC\_vli\_add}} \cdot P_{\text{uECC\_vli\_add}}) \\ & + (N_{\text{uECC\_vli\_sub}} \cdot P_{\text{uECC\_vli\_sub}}) \\ & + P_{\text{RNG}} + P_{\text{Tx}} + P_{\text{Rx}} + P_{\text{Listen}} + P_{\text{Overhead}} \end{aligned} \quad (2)$$

The implementation of Curve25519[2] has a different set of APIs. The adapted version is given in Equation (3).

$$\begin{aligned} P_{\text{ECDH\_curve25519\_total}} = {} & (N_{\text{fe25519\_square}} \cdot P_{\text{fe25519\_square}}) \\ & + (N_{\text{square256\_asm}} \cdot P_{\text{square256\_asm}}) \\ & + (N_{\text{fe25519\_mul}} \cdot P_{\text{fe25519\_mul}}) \\ & + (N_{\text{multiply256x256\_asm}} \cdot P_{\text{multiply256x256\_asm}}) \\ & + (N_{\text{fe25519\_mpyWith121666\_asm}} \cdot P_{\text{fe25519\_mpyWith121666\_asm}}) \\ & + (N_{\text{fe25519\_add}} \cdot P_{\text{fe25519\_add}}) \\ & + (N_{\text{fe25519\_sub}} \cdot P_{\text{fe25519\_sub}}) \\ & + P_{\text{RNG}} + P_{\text{Tx}} + P_{\text{Rx}} + P_{\text{Listen}} + P_{\text{Overhead}} \end{aligned} \quad (3)$$

Some of the salient features that can be observed is that 96 % of the work load is caused by the uECC_vli_add API for NIST curves. The API multiply256x256_asm for Curve25519 accounts for more than 99 % of the energy consumption. Also, Curve25519 implementation has a constant function call count irrespective of the generated random number.

The statistics for curve secp256r1 and curve25519 are given in tables 1 and 2, respectively. They show the number of API calls and the corresponding energy consumption for generating a single shared secret key. We implemented and measured all NIST curves, but give details only on secp256r1, because it provides the same security as curve25519 and is currently *de facto* standard. Total energy costs are given in Table 3.

[1]https://github.com/kmackay/micro-ecc
[2]https://cr.yp.to/ecdh.html

Table 1: Average energy consumption statistics for curve Secp256r1.

API name	# API calls	Energy per API call (μJ)	Total energy per key (μJ)
vli_mmod_fast	7216	0.004	26.369
uECC_vli_add	54986	0.666	36637.700
uECC_vli_sub	39548	0.004	177.381
uECC_vli_mult	4642	0.010	48.460
uECC_vli_square	2574	0.007	17.535
uECC_vli_clear	8	0.018	0.141
uECC_vli_set	8265	0.001	10.053
uECC_vli_cmp_unsafe	4548	0.001	4.555
uECC_vli_rshift1	1446	0.002	3.041
uECC_vli_modSquare_fast	2574	0.001	1.207
Wi-Fi Tx	1	20.767	20.767
Wi-Fi Rx	1	21.005	21.005
Sum	125809		**36965.214**

Table 2: Average energy consumption statistics for Curve25519.

API name	# API calls	Energy per API call (μJ)	Total energy per key (μJ)
fe25519_sub	2040	0.028	57.242
fe25519_add	2040	0.005	9.994
fe25519_mul	2574	0.001	2.462
multiply256x256_asm	2574	5.240	13486.839
fe25519_square	2548	0.004	10.640
square256_asm	2548	0.007	16.816
fe25519_reduceTo256Bits_asm	5122	0.001	3.690
fe25519_cswap	1024	0.002	2.088
fe25519_mpyWith121666_asm	510	0.001	0.753
Wi-Fi Tx	1	20.767	20.767
Wi-Fi Rx	1	21.005	21.005
Sum	20980		**13632.296**

Table 3: Total energy consumption for ECDH curves measured including Wi-Fi communication. The bold curves have an equivalent security of 128 bit as our CBKA key generation.

ECDH Curve	Total energy per key (mJ)
Secp160r1	18.540
Secp192r1	16.796
Secp224r1	21.541
Secp256r1	**36.970**
Secp256k1	39.983
Curve25519	**13.630**

Alice and Bob need to exchange just one packet of data for exchanging their public key, while the curve domain parameters can be fixed. So, the total energy required to generate a single shared secret key in case of Secp256r1 is 36.97 mJ and for Curve25519 13.63 mJ.

4.2 Channel-Based Key Agreement

As explained in Section 2, the energy consumption of CBKA is highly dependent on the entropy source's quality, i.e., the channel measurements. The BDR depends on the reciprocity of the channel. In addition to the reciprocity, the channel should also be dynamic, i.e., the measured channel characteristics should have a reasonable amount of variation, or there will not be enough entropy available in the measured channel properties [28]. We extract RSSI values from ping-pong messages to measure the channel.

We implemented different quantization schemes to convert a raw RSSI sequence to a bit string. These are Mathur Single Bit [17], Mathur Multi Bit [17], Jana Multi Bit [14] and Ambekar Multi Bit [1]. As part of some quantization schemes, a single packet of data needs to be exchanged between Alice and Bob for alignment. In Table 4, we give the results of our measured quantization implementation. The number of output bits can vary, so we give an average number for a mobile scenario and a theoretical optimum.

One information reconciliation scheme we use is the code-offset construction [6] based on a $(n = 127, k = 64, t = 10)$-BCH code [10]. With this setting, a BDR of up to 8 % can be successfully reconciled and the number of leaked bits is fixed to $n - k = 63$ per block. The second reconciliation scheme is BBBSS [2], where the number of exchanged bits depend on the BDR. Every transmitted bit leaks one bit of entropy. The number of disclosed bits n_d is given by Bennett *et al.* [2] with $n_d/n_0 \approx (1.1 + e) \cdot h(e)$, where e is the BDR and $h(e) = -e \cdot \log_2(e) - (1-e) \cdot \log_2(1-e)$. When e becomes as large as about 20 %, BBBSS discloses all the information in the bit sequence and becomes unusable. Table 5 provides details about the reconciliation schemes for different BDR. It includes the rates of successful reconciliation, the number of disclosed bits, and the computational as well as the total energy per reconciled string.

For online entropy estimation we use Lempel-Ziv-Welch compression [25]. It is a universal lossless data compression algorithm. The idea is that full entropy data cannot be further compressed. So whenever a compression is possible, the inherent entropy is limited by this compression rate.

Table 4: Computational energy consumption for different quantization schemes for a single block. Each scheme quantizes an input block of 256 measurements.

Quantization scheme	Output bits (Average / best case)	Comp. energy per block (μJ)	# exchanged packets	total energy per block (μJ)
Mathur Single Bit	145 / 256	504.91	1	546.68
Mathur Multi Bit	421 / 512	699.58	1	741.36
Jana Multi Bit	512 / 512	5349.19	0	5349.19
Ambekar Multi Bit	512 / 512	724.89	0	724.89

Table 5: Computational energy consumption for BCH and BBBSS information reconciliation schemes for a single block as a function of BDR. Both schemes have a block length of 127 bit. Also the success rate and disclosed bits per block per scheme are given.

BDR	BBBSS				BCH-based			
	success rate	disclosure (bits)	computational energy per block (μJ)	total energy per block (μJ)	success rate	disclosure (bits)	computational energy per block (μJ)	total energy per block (μJ)
0 %	100 %	8	198.17	365.26	100 %	63	946.36	988.13
1 %	100 %	20	306.85	578.37	100 %	63	1190.92	1232.69
2 %	97.10 %	29	298.01	673.96	100 %	63	1204.78	1246.55
3 %	98.70 %	37	344.33	824.71	99.85 %	63	1218.25	1260.02
4 %	96.60 %	44	395.43	980.24	98.74 %	63	1245.89	1287.66
5 %	92.60 %	51	416.28	1105.52	94.75 %	63	1132.93	1174.70
6 %	88.30 %	57	439.58	1233.25	85.86 %	63	1268.87	1310.64
7 %	81.70 %	63	453.75	1351.85	72.07 %	63	1281.61	1323.38
8 %	64.20 %	69	476.59	1479.12	55.21 %	63	1302.39	1344.16

Table 6: Energy consumption for entropy estimation and privacy amplification stages.

CBKA step	Energy (μJ)
Entropy Estimation	130.90
Privacy Amplification	116.83

Using lossless compression for entropy estimation is part of other research, e.g. [13].

Once entropy estimation is completed, the last step is privacy amplification with a cryptographic hash function and its energy consumption is constant. The measured results for privacy amplification and entropy estimation are given in Table 6. Equation (4) combines all of the previous steps for a single key via CBKA.

$$\begin{aligned} P_{\mathrm{CBKA}} = {} & N_{\mathsf{CM}} \cdot ((P_{\mathrm{Tx}} + P_{\mathrm{Rx}} + P_{\mathrm{Listen}}) \cdot T_{\mathrm{RTT}}) \\ & + N_{\mathsf{Q}} \cdot (P_{\mathsf{Q}} + (P_{\mathrm{Tx}} + P_{\mathrm{Rx}} + P_{\mathrm{Listen}}) \cdot T_{\mathrm{RTT}})) \\ & + N_{\mathsf{IR}} \cdot (P_{\mathsf{IR}} + (P_{\mathrm{Tx}} + P_{\mathrm{Rx}} + P_{\mathrm{Listen}}) \cdot T_{\mathrm{RTT}})) \\ & + N_{\mathsf{EE}} \cdot P_{\mathsf{EE}} \\ & + P_{\mathsf{PA}} \\ & + P_{\mathrm{Overhead}}, \end{aligned} \tag{4}$$

where N_i represents the number of times each CBKA step i is executed, P_i represents the *computational* energy consumption per call i and i represents an element from the set of CBKA steps. This set consists of Channel Measurement (CM), Quantization (Q), Information Reconciliation (IR), Entropy Estimation (EE) and Privacy Amplification (PA). T_{RTT} is the round trip time for a ping-pong message used for CM, which is also the time needed for the Wi-Fi module to be turned on as described in Section 3. The energies for transmitting P_{Tx}, receiving P_{Rx} and listening P_{Listen} are the same as for ECDH.

5. COMPARISON OF ECDH AND CBKA

Wi-Fi is a major contributor to the total energy consumption of channel-based key agreement. The number of packets needed to be exchanged for deriving a 128-bit full entropy key, including the packets required for channel measurements and the packets for the different schemes of CBKA, are also shown in Table 7. The number of needed channel measurements ranges from 1500 to 2300 for the considered practical scenario. Contrary, Zenger *et al.* [28] report 5396 of needed measurements in a mobile scenario.

Comparing the energy consumption based on the number of needed packets from our implementation and in our mobile scenario, the required energy for CBKA is much higher than ECDH. A good CBKA scenario still consumes more than seven times of the energy for key generation than ECDH. The total required energy for ECDH is given in Table 3 and for CBKA in Table 7. A direct comparison between CBKA and ECDH is given in Table 8, alongside the improvement, which we introduce in Section 6.

For a practical application, the number of quantized bits per RSSI block varies and so does the required number of exchanged packets of the CBKA algorithm. Also, the estimated entropy varies with the channel measurements and number of disclosed bits during reconciliation. Because of this, we measured different quantization and reconciliation procedures. Considering a differing amount of RSSI measurements for a key generation process, we measured combinations of quantization and reconciliation schemes, as shown in Table 7. Obviously, the channel measurement dominates the overall energy consumption significantly for every combination of quantization and reconciliation. For determining a lower bound on the energy consumption we introduce the following assumptions:

- Each quantizer outputs the maximum of bits.
- All measurement blocks can be reconciled successfully.

Table 7: Energy consumption for a full entropy 128-bit key with various quantization and information reconciliation schemes.

CBKA step	Wi-Fi / computational energy (mJ)	practical/ optimal scenario	Mathur Single Bit		Mathur Multi Bit		Jana Multi Bit	
			BBBSS	BCH-based	BBBSS	BCH-based	BBBSS	BCH-based
CM	# needed measurements	practical	2048	1792	1792	1536	1536	2304
		optimal	256	256	128	128	128	128
CM	Wi-Fi	practical	85.55 mJ	74.86 mJ	74.86 mJ	64.16 mJ	64.16 mJ	96.24 mJ
		optimal	10.69 mJ	10.69 mJ	5.35 mJ	5.35 mJ	5.35 mJ	5.35 mJ
Q	computational	practical	4.04 mJ	3.53 mJ	4.90 mJ	4.20 mJ	32.10 mJ	48.14 mJ
		optimal	0.50 mJ	0.50 mJ	0.35 mJ	0.35 mJ	2.67 mJ	2.67 mJ
	Wi-Fi	practical	0.33 mJ	0.29 mJ	0.29 mJ	0.25 mJ	0.25 mJ	0.38 mJ
		optimal	0.04 mJ	0.04 mJ	0.02 mJ	0.02 mJ	0.02 mJ	0.02 mJ
IR	computational	practical	4.32 mJ	10.41 mJ	10.97 mJ	25.90 mJ	11.44 mJ	47.26 mJ
		optimal	0.95 mJ	2.63 mJ	0.95 mJ	2.63 mJ	0.95 mJ	2.63 mJ
	Wi-Fi	practical	9.09 mJ	0.33 mJ	23.08 mJ	0.83 mJ	24.06 mJ	1.52 mJ
		optimal	2.01 mJ	0.08 mJ	2.01 mJ	0.08 mJ	2.01 mJ	0.08 mJ
EE	computational	practical	0.65 mJ	0.52 mJ	1.57 mJ	1.31 mJ	1.57 mJ	2.49 mJ
		optimal	0.13 mJ	0.13 mJ	0.13 mJ	0.13 mJ	0.13 mJ	0.13 mJ
PA	computational	practical	0.12 mJ	0.12 mJ	0.12 mJ	0.12 mJ	0.12 mJ	0.12 mJ
		optimal	0.12 mJ	0.12 mJ	0.12 mJ	0.12 mJ	0.12 mJ	0.12 mJ
All	**total energy per key**	practical	**112.46 mJ**	**104.01 mJ**	**131.66 mJ**	**126.87 mJ**	**177.23 mJ**	**291.53 mJ**
		optimal	**15.90 mJ**	**17.33 mJ**	**10.23 mJ**	**11.65 mJ**	**14.88 mJ**	**16.30 mJ**

- The measured channel characteristics have full entropy, due to dynamic behavior between Alice and Bob.

This means that for a lower bound estimation we use Equation (4) with minimal parameters N_{CM}, N_{Q}, N_{IR} and N_{EE}. We state the lower bound on the energy consumption for CBKA also in Table 7.

For a lower bound estimation we need to consider schemes with the lowest energy consumption. In Figure 3, we illustrate the number of needed channel measurements compared to the total energy consumption for one key. In this figure we assume that a key generation is possible with a limited amount of measurements, e.g., with increasing inherent entropy. As a reference, the energy consumption of secp256r1 and curve25519 is also shown in the figure. Note that the intersection point is also a break even point of the amount of packets need to be exchanged in order to generate a single key at the same energy consumption level as that of the two ECDH algorithms. We also estimated a more efficient implementation with values we found in the literature. The implementation by Zenger *et al.* consumes less energy. They state $6\,\mu$J for their cheapest quantizer, $290\,\mu$J for reconciliation and $6\,\mu$J for privacy amplification. Figure 3 shows four key generation algorithms in total – curve25519, secp256r1 and two implementations of CBKA. Each of these algorithms is illustrated with its computational energy and its total energy, where the energy of Wi-Fi is included. As one can see, the communication energy of ECDH algorithms are negligible. On the contrary, the energy gap between the computational and total energy for CBKA algorithms increases with an increased number of exchanged packets.

In order to achieve a smaller (or equal) energy consumption of CBKA than curve25519 for a single key generation, CBKA must use at most 304 packets for channel measurements, quantization and reconciliation. Compared to curve secp256r1, CBKA may only utilize up to 835 packets to be cheaper in terms of energy consumption.

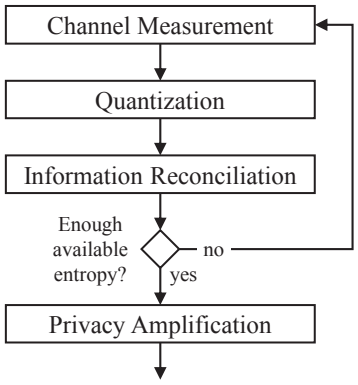

Figure 4: Well known process for extracting secret keys from a communication channel.

6. ENERGY IMPROVEMENT FOR CBKA

In the following, we assume the nodes can access a common source of randomness C_{CR}. In order to generate a common secret key, they follow a process similar to the one shown in Figure 4, which is also described in Section 2. As the BDR between samples is not known beforehand, neither how many can be reconciled and how much information has to be disclosed to a potential attacker Eve in order to correct existing mismatches, it cannot be precisely predicted how many data points have to be captured so a key of specific length can be generated. Consequently, an unspecified amount of data has to be collected during channel measurement. The actual amount of secret key material is not known until information reconciliation and entropy estimation is completed. If the required sequence length has not been achieved, the process has to be continued by another iteration etc. However, if this scheme is repeated for some iterations, it is likely to surpass the required entropy, such that more data has been collected than would have been re-

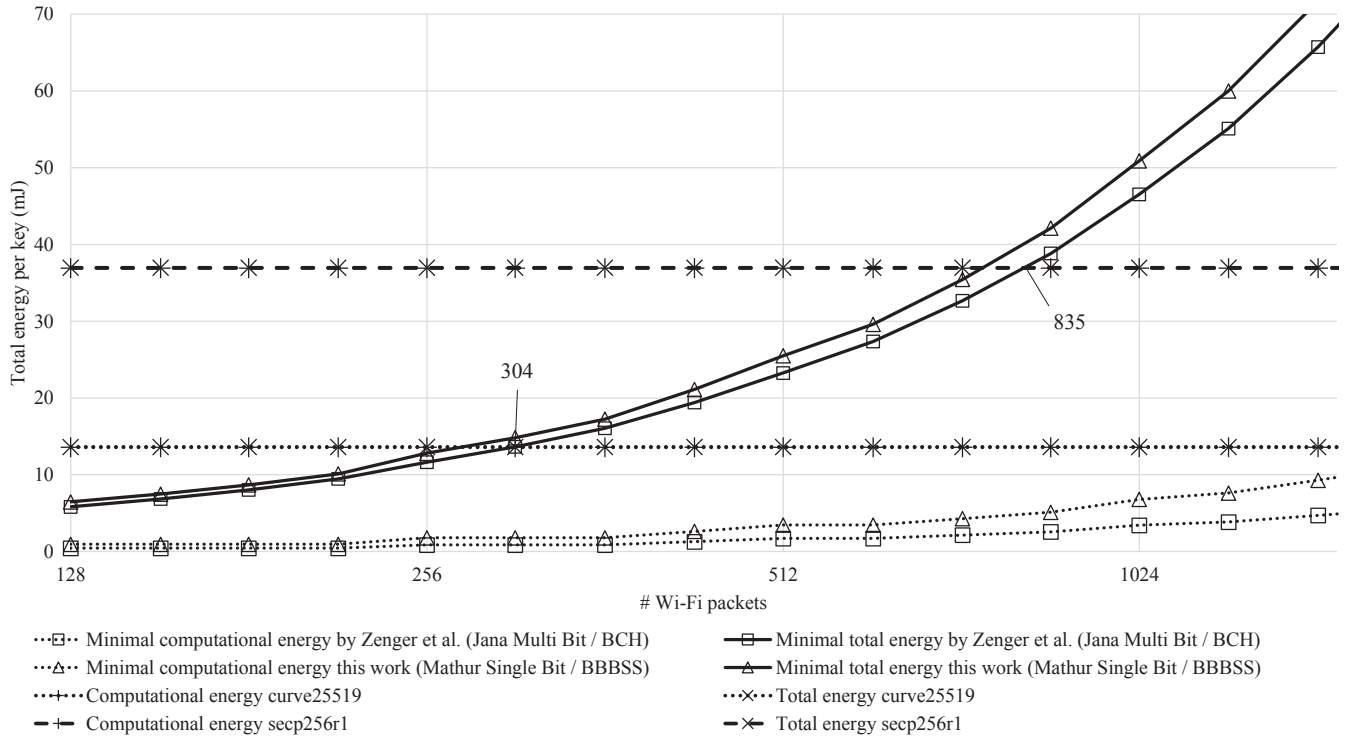

Figure 3: Lower bound of total energy consumption per key as a function of number of Wi-Fi packets. The figure shows that CBKA consumes less energy than ECDH (curve25519) if no more than 304 packets are transmitted via Wi-Fi, or no more than 835 packets for secp256r1. While the energy gap for CBKA between computational and total energy increases with more Wi-Fi packets, it remains negligible small and constant for ECDH. We include the results of Zenger *et al.* [28] and the results of this work. Note that this graph shows the very optimal case. In a good scenario, roughly 1500 to 5000 channel measurements are needed for a 128-bit key.

quired. In terms of energy consumption, this overhead can be costly, as we have shown in the previous sections.

We approach this issue by extending the key generation process as it is simplified in Figure 5. The idea comprises to exploit data packets transmitted during information reconciliation in order to additionally probe the communication channel, thus making one iteration of this process more efficient since more data can be collected.

6.1 Improved Protocol

The proposed, iterative protocol is shown in detail in Figure 6. Here we use the same nomenclature as in the work of Huth *et al.* [11]. Alice and Bob both have access to a common source of randomness C_{CR}, which they probe and quantize, with a channel measurement function CM and quantizer Q respectively at time t. This results in similar quantized values q. Next, Alice generates some redundancy information r with her reconciliation funtion IR. Upon receiving Alice's r, Bob can reconcile his quantized values to those of Alice with his version of the reconciliation function IR', so that both have the same values of q.

For reciprocal channel measurements two messages transmitted within the coherence time are required. The idea of the proposed protocol is to precalculate the redundancy information on Alice's side, so she can respond to Bob's messages without delay. The exchanged messages *ack* and r can be used for channel measurements.

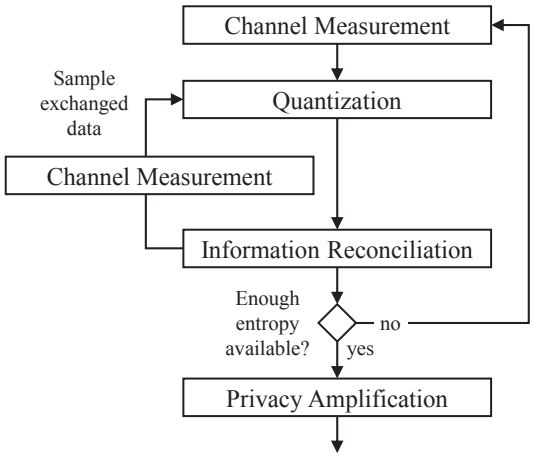

Figure 5: Extended secret key extraction process.

Our protocol can be adapted for information reconciliation as BBBSS and BCH-based alike. In the case of BCH-based reconciliation, $r := \mathsf{IR}(q)$ represents the calculation of a sketch r, as in [6]. The corresponding recover procedure is written as $q := \mathsf{IR}'(q', r)$, which inputs a similar q' and the received sketch r. The reconciliation succeeds if the

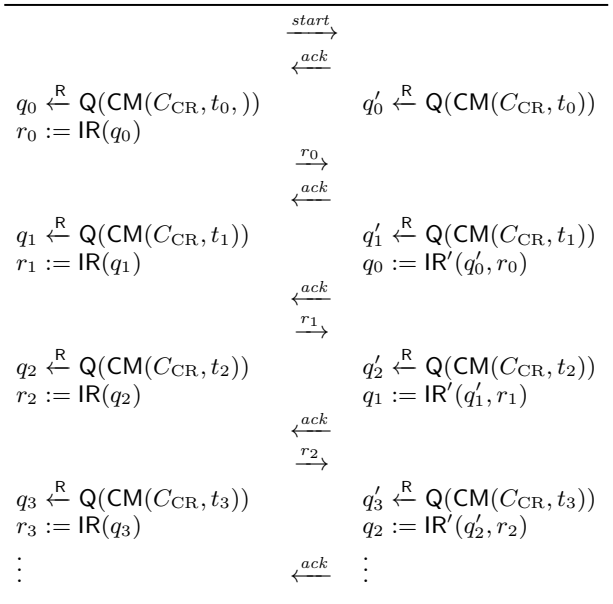

Figure 6: General idea of how the transmitted redundancy during information reconciliation can be used for channel measurements. We denote the common source of randomness with C_{CR}, time t, quantized measurements q and transmitted redundancy r.

distance between q and q' is no greater than the error correcting capabilities of the used BCH code. The message *ack* indicates that Bob is finished with decoding and is ready to receive another sketch. Note, that the recover part of a secure sketch is computationally more intense resulting in a longer execution time as the sketching part.

In case of BBBSS [2], $r := \mathsf{IR}(q)$ represents the calculation of a parity from the bisecting algorithm. The bisecting algorithm is written as $q := \mathsf{IR}'(q', r)$, which inputs a similar q' and the received parity bits r. The message *ack* represents the bit positions where Bob has disagreeing parities. The reconciliation protocol BBBSS especially benefits from our adaption. It exchanges a lot of parity bits, whereas a secure sketch only transmits one message per reconciliation.

It shall be noted that encoding and decoding, depending on the type of coding or reconciliation scheme, are usually not of equivalent computational effort. Hence, in a heterogeneous setup, e.g., with star topology, where the central node is represented by a powerful base station, that communicates with one or more resource-constraint devices. Protocol 6 takes advantage of this and allows one node to always encode and the other node to always decode. Therefore, the protocol can be adjusted to the computational constraints of a certain system. Nevertheless, as some systems, e.g. in homogeneous setups, do not have such constraints or even require a more balanced computational complexity, we can also change the protocol. Here, in one iteration encoding and decoding is done by just one node (say Bob), and done by the other node (Alice) in the next iteration, so that the roles of Alice and Bob change after each reconciliation round.

6.2 Improvement Results

The protocol proposed in Figure 6 offers several benefits. First, the measurement of the communication channel (or some property depending on this channel) is interleaved with the reconciliation process. By exploiting communication overhead from reconciliation protocols, additional channel probing can be performed. So, in best case, there is no designated channel measurement needed anymore which potentially cannot be used for other functionalities. Second, due to the iterative design of the protocol, it is possible to calculate exactly if another send/receive step is needed (in terms of successfully generated key material), providing the possibility of saving energy by a decreased number of required transmissions. Third, computationally more challenging tasks, like decoding, can be assigned to the more powerful communication partner. And at last, due to a higher number of transmissions, interactive reconciliation protocols, e.g. BBBSS [2] and Cascade [4], have been understood to be costly for resource-constraint devices. Our protocol exploits these additional transmissions in an useful manner and, hence, makes these more energy effective for such devices, as BBBSS and Cascade need less computational energy than, e.g., a BCH code.

We adapted our proposed protocol to our previously taken measurements and the results are shown in Table 8. It presents the energy improvement of two different reconciliation methods for three scenarios, where a good scenario has a lot of movement of the devices involved and a bad scenario is nearly static. As previously discussed, BBBSS shows a good energy improvement of up to 25 %, whereas BCH-based reconciliation can only be improved by up to 2 %. Overall, we need 116 mJ to 400 mJ for a single key using CBKA. Recall, that key generation with ECDH is much cheaper, where the energy ranges from 13.63 mJ to 36.97 mJ depending on the curve.

7. DISCUSSION

Considering the most used curves for ECDH, secp256r1 (when using the micro-ECC library implementation) consumes 36.97 mJ and curve25519 (when using the μNaCl library implementation) consumes 13.63 mJ for generating a single shared secret, including Wi-Fi communication.

In case of ECDH NIST Curves, the majority of workload happens in the uECC_vli_add API. Measuring just this API gives a good overall energy estimate since it consumes 95 % of the computational energy. The domain curve parameters exchange between two entities accounts only for few bytes (only a single packet in terms of network communication) and results in a constant energy consumption. For curve25519 alike, there is one API which contributes 99 % of the computational energy – multiply256x256_asm. Therefore, for future optimizations the uECC_vli_add and multiply256x256_asm APIs would be an excellent starting point to reduce the overall energy consumption.

For CBKA, the energy consumption of the data transmission over Wi-Fi plays a critical part in the overall energy consumption for generating a single shared secret. We considered different quantization and information reconciliation schemes and measured the energy consumption from an experimental scenario, resulting in an energy range from 116.02 mJ to 291.53 mJ for a single key. Note, that the most expensive part of the key generation is the channel measure-

Table 8: Table of our improved results, where exchanged information reconciliation messages get probed for channel measurements. Each practical scenario show a needed number of packets to derive a 128-bit key, e.g., 1500 packets are needed in a good scenario. The improvement (written as old energy → new energy) with BBBSS as reconciliation is up to 25 %, for BCH up to 2 %. As a reference we also give the required energy of a ECDH key from Table 3.

practical scenario (# packets needed)	Energy per key CBKA with BBBSS	Energy per key CBKA with BCH	Energy per key Secp256r1	Energy per key Curve25519
good (ca. 1500)	154.90 mJ → 116.02 mJ	155.70 mJ → 154.22 mJ	36.97 mJ	13.63 mJ
medium (ca. 2300)	238.16 mJ → 181.71 mJ	239.39 mJ → 237.13 mJ		
bad (ca. 5000)	528.53 mJ → 398.93 mJ	531.20 mJ → 520.89 mJ		

ments, i.e., the establishment of the channel. When channel measurements can be assumed for free, then CBKA is always cheaper than ECDH, as depicted in Figure 3. So we conclude that for an initial key generation ECDH could be used and for rekeying CBKA, when the occuring communication is also used for channel probing.

Zenger *et al.* [28] also worked on the energy consumption for different protocol layers of CBKA. They measured for a key generation on the curve secp384r1 a total energy of 115.69 mJ. However, we focus on curves providing 128-bit security for IoT applications, so that the curves are comparable to CBKA. If we estimate the energy consumption with Equation (4), CBKA would need 8.5 mJ for an optimal and 157 mJ for an average scenario for the same security as secp384r1.

8. CONCLUSION

When Wi-Fi is used as the wireless communication interface for IoT devices, our results show that CBKA algorithms consume more energy to generate a single key, compared to ECDH algorithms. We also acknowledge that CBKA code implementations are relatively young. Optimizations could reduce the energy consumption for channel measurement, quantization and information reconciliation. The computational portion of CBKA consumes the least amount of energy and data exchange over Wi-Fi is the major contributing factor for the overall energy consumption.

In summary, when we assume that the channel measurements are not free in terms of energy consumption for an IoT device with Wi-Fi interface, then CBKA consumes more energy than ECDH, unless the key can be generated with exchange of

- less than 304 packets, compared to curve25519 and
- less than 835 packets, compared to secp256r1.

We remark these numbers are very optimistic, as a practical mobile scenario requires around 1500 to 5000 packets of data exchange before a key can be derived.

9. REFERENCES

[1] A. Ambekar, M. Hassan, and H. D. Schotten. Improving channel reciprocity for effective key management systems. In *Signals, Systems, and Electronics (ISSSE), 2012 International Symposium on*, pages 1–4. IEEE, 2012.

[2] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. Experimental quantum cryptography. *Journal of cryptology*, 5(1):3–28, 1992.

[3] D. J. Bernstein. Curve25519: new diffie-hellman speed records. In *Public Key Cryptography-PKC 2006*, pages 207–228. Springer, 2006.

[4] G. Brassard and L. Salvail. Secret-key reconciliation by public discussion. In *advances in Cryptology - EUROCRYPT'93*, pages 410–423. Springer, 1993.

[5] G. De Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,*, pages 580–585. IEEE, 2008.

[6] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in cryptology-Eurocrypt 2004*, pages 523–540. Springer, 2004.

[7] S. Gopinath, R. Guillaume, P. Duplys, and A. Czylwik. Reciprocity enhancement and decorrelation schemes for phy-based key generation. In *Globecom Workshops (GC Wkshps), 2014*, pages 1367–1372, Dec 2014.

[8] R. Guillaume, S. Ludwig, A. Müller, and A. Czylwik. Secret key generation from static channels with untrusted relays. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on*, pages 635–642, Oct 2015.

[9] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic hardware and embedded systems-CHES 2004*, pages 119–132. Springer, 2004.

[10] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres (paris)*, 2(147-156):116, 1959.

[11] C. Huth, R. Guillaume, T. Strohm, P. Duplys, I. A. Samuel, and T. Güneysu. Information reconciliation schemes in physical-layer security: A survey. *Computer Networks*, 2016.

[12] C. Huth, J. Zibuschka, P. Duplys, and T. Güneysu. Securing systems on the internet of things via physical properties of devices and communications. In *Systems Conference (SysCon), 2015 9th Annual IEEE International*, pages 8–13. IEEE, 2015.

[13] T. Ignatenko, G.-J. Schrijen, B. Skoric, P. Tuyls, and F. Willems. Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method. In *Information Theory, 2006 IEEE International Symposium on*, pages 499–503. IEEE, 2006.

[14] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera,

N. Patwari, and S. V. Krishnamurthy. On the effectiveness of secret key extraction from wireless signal strength in real environments. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 321–332. ACM, 2009.

[15] S. Kumar, M. Girimondo, A. Weimerskirch, C. Paar, A. Patel, and A. S. Wander. Embedded end-to-end wireless security with ecdh key exchange. In *Circuits and Systems, 2003 IEEE 46th Midwest Symposium on*, volume 2, pages 786–789. IEEE, 2003.

[16] S. S. Kumar. *Elliptic curve cryptography for constrained devices*. PhD thesis, Ruhr University Bochum, 2006.

[17] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 128–139. ACM, 2008.

[18] S. NIST. Special publication 800-22. *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, 2010.

[19] N. Patwari, J. Croft, S. Jana, and S. K. Kasera. High-rate uncorrelated bit extraction for shared secret key generation from channel measurements. *Mobile Computing, IEEE Transactions on*, 9(1):17–30, 2010.

[20] A. Rice and S. Hay. Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive and Mobile Computing*, 6(6):593–606, 2010.

[21] Silabs. Starter kit EFM32GG-STK3700 User Manual. https://www.silabs.com/Support%20Documents/TechnicalDocs/efm32gg-stk3700-ug.pdf.

[22] M. A. Tope and J. C. McEachen. Unconditionally secure communications over fading channels. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 1, pages 54–58. IEEE, 2001.

[23] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328. IEEE, 2005.

[24] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305, 2002.

[25] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[26] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. The design and use of simplepower: A cycle-accurate energy estimation tool. In *Proceedings of the 37th Annual Design Automation Conference*, DAC '00, pages 340–345, New York, NY, USA, 2000. ACM.

[27] C. T. Zenger, J. Förster, and C. Paar. Implementation and evaluation of channel-based key establishment systems, 2014.

[28] C. T. Zenger, J. Zimmer, M. Pietersz, J.-F. Posielek, and C. Paar. Exploiting the physical environment for securing the internet of things. In *Proceedings of the 2015 New Security Paradigms Workshop*, pages 44–58. ACM, 2015.