2019

Modeling Context-Adaptive Energy-Aware Security in Mobile Devices

Preeti Singh

University of North Florida

MODELING CONTEXT-ADAPTIVE ENERGY-AWARE SECURITY IN MOBILE DEVICES

by

Preeti Singh

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

April, 2019

The thesis "Modeling Context-Adaptive Energy-Aware Security in Mobile Devices" submitted by Preeti Singh in partial fulfillment of the requirements for the degree of Master of Science in Computing and Information Sciences has been

Approved by the thesis committee: Date:

Dr. Swapnoneel Roy
Thesis Advisor and Committee Chairperson

Dr. Asai Asaithambi

Dr. Roger E. Eggen

Accepted for the School of Computing:

Dr. Sherif Elfayoumy
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Dr. William F. Klostermeyer
Interim Dean of the College

Accepted for the University:

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEGEMENTS

I would like to take the opportunity to sincerely thank all the people who helped me during the completion of this thesis. I am grateful to my thesis advisor, guide and motivator Dr. Swapnoneel Roy, who has been a source of inspiration in the completion of this research work.

This thesis is a result of his guidance and support. I would like to thank my thesis committee members, Dr. Asai Asaithambi and Dr. Roger E. Eggen, for providing insightful comments during my prospectus presentation which helped me address the topic more broadly.

I would also like to thank my family members who supported me in whatever ways they could in the completion of this work.

Last but not the least, I would also like to thank each faculty member in the UNF School of Computing for always being helpful and motivating me to complete this thesis research and my Master's Degree.

CONTENTS

List of Figures vii

List of Tables viii

Abstract ix

Chapter 1 INTRODUCTION 1

1.1 Background 3

1.2 An Example 6

1.3 Previous Work 7

Chapter 2 NEW ALGORITHMS FOR CONTEXT-SEC 9

2.1 NP-Completeness of CONTEXT-SEC 9

2.2 New Algorithms for solving CONTEXT-SEC 10

2.2.1 An Optimal Dynamic Programming Algorithm 11

2.2.2 A Greedy Algorithm 14

2.2.3 A Fully Polynomial Time Approximation Scheme (FPTAS). 16

2.3 Comparison of Computational Complexities 17

Chapter 3 IMPLEMENTATION OF THE ALGORITHMS. 18

3.1 Implementing the Dynamic Programming Algorithm 18

3.2 Implementing the Greedy Algorithm 22

3.3 Implementing the FPTAS Algorithm 23

3.4 Data Structures Used 25

Chapter 4 RESULTS AND DISCUSSION 26

4.1 Data-Sets 26

4.2 Results. 28

4.3 Discussion 30

Chapter 5 CONCLUSION AND FUTURE WORK 31
5.1 Conclusion. 31
5.2 Future Work 31
REFERENCES 32
LIST OF PUBLICATIONS FROM THE THESIS. 37
Vita. 38

FIGURES

Figure 3.1 Final solution of Algorithm 1 21

Figure 3.2 Final solution of Algorithm 2 23

Figure 3.3 Final solution of Algorithm 3 25

Figure 4.1 Solution values produced for the two data-sets. 29

Figure 4.2 Execution Time in seconds. 29

TABLES

Table 1.1	Multi-level Security [28]	6
Table 1.2	Energy and Performance Evaluation [28]	7
Table 2.1	Comparison of Runtime Complexity	13
Table 2.2	Comparison of the three algorithms.	17
Table 3.1	Table for $i = 1$	20
Table 3.2	Table for $i = 7$	20
Table 3.3	Table for $i = 26$	21
Table 4.1	Data-Sets	27

ABSTRACT

As increasing functionality in mobile devices leads to rapid battery drain, energy management has gained increasing importance. However, differences in user's usage contexts and patterns can be leveraged for saving energy. On the other hand, the increasing sensitivity of users' data, coupled with the need to ensure security in an energy-aware manner, demands careful analyses of trade-offs between energy and security. The research described in this thesis addresses this challenge by: 1) modeling the problem of context-adaptive energy-aware security as a combinatorial optimization problem (CONTEXT-SEC); 2) proving that the decision version of this problem is NP-Complete, via a reduction from a variant of the well known KNAPSACK problem; 3) developing three different algorithms to solve a relaxed offline version of CONTEXT-SEC; and 4) implementing tests and compares the performance of the above three algorithms with data-sets derived from real-world smart-phones on wireless networks.

The first algorithm presented is a pseudo-polynomial dynamic programming (DP) algorithm that computes an allocation with optimal user benefit using recurrence of the relations; the second algorithm is a greedy heuristic for allocation of security levels based on user benefit per unit of power consumption for each level; and the third algorithm is a Fully Polynomial Time Approximation Scheme (FPTAS) which has a polynomial time execution complexity as opposed to the pseudo-polynomial DP based approach. To the best of the researcher's knowledge, this is the first work focused on modeling, design, implementation and experimental performance

analysis of any algorithm for context-adaptive energy-aware security. The results will be useful for researchers and practitioners working in this area.

CHAPTER 1

INTRODUCTION

Energy management has become one of the foremost concerns in the design of mobile devices, due primarily to the increasing functionality of mobile applications which rapidly drain battery power in these devices. The computation-intensive cryptographic operations involved in securing communications significantly contribute to energy drain. Current approaches to security often do not cater to varying user context (for example, varying security levels depending on the user's current location), resulting in inefficient energy usage by mobile devices. Furthermore, current approaches do not adapt to energy-security trade-offs that vary for individual users. Research reveals that energy use patterns are unique and driven by context (e.g. location of a user) [2, 3, 13, 31].

These considerations motivate the need to develop energy-aware security mechanisms for mobile devices. Tailoring security to user's context can facilitate efficient use of energy, thus enabling the device to meet a given power budget. Hence, context-adaptive security designs are a promising approach for facilitating efficient energy usage by a mobile device while meeting its desired security level requirements.

Context-adaptive security designs must adapt security level and also prioritize allocation of resources (e.g. energy consumption) for mobile devices. Current

approaches to allocation are often static and do not scale resources according to user needs. Approaches are needed for maximizing user utility subject to power constraints. Thus algorithms must be developed for optimizing user utility for a given power budget. Such an approach can facilitate efficient allocation of resources including energy usage.

In this work, context-adaptive energy-aware security is modeled for mobile devices as a combinatorial optimization problem and solutions are developed for effective allocation of resources (e.g. security levels) such that user utility is maximized. In particular, the decision version of the problem has been proved to be NP-Complete and three different algorithms to the problem have been designed and experimentally tested for the problem. The major contributions of this work is summarized below:

- Modeling and formulating the problem as a combinatorial optimization problem;
- Showing that the decision version of the problem is NP-Complete through a reduction from k-value Knapsack;
- Designing a pseudo-polynomial optimal dynamic programming algorithm to solve the problem;
- Designing a greedy algorithm that maximizes user benefit while meeting a given power budget constraint;
- Designing an FPTAS for the problem which improves upon the execution time of the dynamic programming algorithm;
- Implementing and testing the three algorithms with a real-world smartphone usage and wireless network data set to compare their performance.

The thesis is organized as follows. This first chapter (Chapter 1) provides some basic terminology, a detailed description of the model and formulation of the problem CONTEXT-SEC, and a brief summary of previous work reported in the literature. Chapter 2 proves the NP-Completeness of CONTEXT-SEC, and presents a detailed description of three new algorithms, which were designed in this work for the problem. Chapter 3 describes the implementation of the algorithms that were developed the data sources for the experiments. Chapter 4 presents the results comparing the performance of the three algorithms on real-world datasets. Finally, Chapter 5 presents some general observations that could expand the scope of this work in the future.

1.1 Background

The real-world scenario that reflects CONTEXT-SEC problem is that of a single mobile user who travels through n different locations. Each location has a preferred security level coming from a set $\mathbf{S}$ of m possible security levels ($\mathbf{S} = \{S_1, S_2, \cdots, S_m\}$). Each security level S_i has a power consumption p_i and a user benefit (in terms of the amount of security) b_i associated with it.

The user, who travels through n locations, is constrained by a total energy budget E, which is the amount of energy the user does not want to exceed during travel. The user also has a target benefit B, which is the minimum total benefit the user wants to achieve during travel to guarantee a certain level of security depending on location.

The energy budget, for example, can be the total time the battery of the user's mobile phone lasts from full charge. The security benefit, for example, can be the number of times the user's mobile phone is in the preferred security level (e.g. home/work/public) during the travel through the locations.

The problem is to find an allocation of security levels (from the set of m security levels) to the n locations through which the user travels, such that the total energy consumed does not exceed E and the benefit is at least B, allocating the preferred security levels to as many locations as possible. With the assumption that there are unlimited copies of each of the m security levels, the same energy level may be allocated to multiple locations.

In the offline version of this model, the user knows beforehand all the n locations along with the preferred security level of each location (security preference parameter). In the online version, the locations and their preferred levels are not known in advance, but are revealed one at a time. In this thesis, an attempt is made to solve the offline version of the problem.

Additionally, the security preference parameter is removed from locations. Therefore the problem is reduced to finding an allocation of security levels (from the set of m security levels) to the n locations the user travels through, such that the total energy consumed does not exceed E and the benefit is at least B, without any preference for a specific security level for any location.

The CONTEXT-SEC problem optimizes security levels based on user location in order to maximize the total user benefit. The decision version of CONTEXT-SEC would be to meet a target total user benefit of B or exceed B, in other words to perform allocations such that the total benefit is $\geq B$, while the total energy consumption is $\leq E$.

We consider the following optimization version of the context-adaptive, energy-aware security problem. For ease of reference, we also named it CONTEXT-SEC.

> CONTEXT-SEC PROBLEM
>
> INPUT: User's energy budget E with power consumed due to security $\{p_1 \cdots p_m\}$ for varying security levels $\{S_1....S_m\} \in \mathbf{S}$ and the corresponding user benefits $\{b_1.....b_n\}$ for n locations through which the user travels.
>
> OUTPUT: For each location i, with $i \in \{1, \cdots, n\}$, allocate a security level of S_{j_i}, where $j_i \in \{1, \cdots, m\}$ to maximize $\sum_{i=1}^{n} b_{j_i}$ subject to $\sum_{i=1}^{n} p_{j_i} \leq E$.

In this thesis the offline version of the CONTEXT-SEC problem is being considered. Additionally, the security preference constraint for locations has not been considered. In other words, in the version of CONTEXT-SEC solved in this thesis, locations do not have preferred security levels. Therefore the problem reduces to having n locations (without any associated preferences) and m security levels to be allocated to those locations such that the total energy consumption is at most E and the total user benefit is maximized.

1.2 An Example

Table 1.1 contains the basis of the proposed multi-level security model for mobile devices. The data displayed in Table 1.1 illustrate a 3-level security solution in which each level is associated with a cryptographic algorithm, as depicted in [28]. A higher security level implies a cryptographic protocol with higher security strength. For instance, AES and RC5, at the low and medium levels respectively, offer authentication but not integrity. On the other hand, HMAC-MD5, at the highest level, offers both authentication and integrity.

Security Level	*Security Property*	*Cryptographic Protocol*
Low	Authentication	AES
Medium	Authentication	RC5
High	Authentication, Integrity Protection	HMAC-MD5

Table 1.1: Multi-level Security [28]

Energy awareness requires that the energy consumption of security protocols be estimated. Table 1.2 presents results from such an estimation of energy along with execution time of security protocols done in [28]. In this work, the researcher uses AvroraZ, a cycle-level power profiling tool, to estimate the execution time and energy consumption of security protocols [28]. As Table 1.2 shows, the energy consumption of mobile devices increases with application of more sophisticated cryptographic protocols for achieving higher security levels.

Cryptographic Protocol	*Execution Time*	*Energy Consumption*
AES	0.2029s	5.435 mJ
RC5	0.3406s	9.124 mJ
HMAC-MD5	4.547s	121.8 mJ

Table 1.2: Energy and Performance Evaluation [28]

The model designed in the work presented in this thesis is generic in that it is not confined to any particular security protocol, but it addresses the broader goal of providing adaptable energy efficient security in mobile devices.

1.3 Previous Work

Prior approaches to security have analyzed the energy consumption of security protocols. The energy consumed by symmetric and asymmetric security protocols in traditional wireless networks is evaluated by Potlapally *et al.* [27] and Hodjat *et al.* [17]. The Internet Protocol Security (IPSec) was optimized to suit the needs of the resource constrained devices by Karri *et al.* [20]. The energy consumed by security protocols within the context of wireless sensor networks was evaluated by Chang *et al.* [7], Ganesan *et al.* [12] and Lee *et al.* [23] .

In addition to energy evaluation, model-based approaches for understanding energy-security tradeoffs have been explored and used by Mckay *et al.* [25]. The authors design a decision-theoretic model to analyze the energy-security tradeoffs in traditional wireless networks in [25]. The tradeoffs in consumption of energy, memory, and security in sensor networks using a key management protocol are analyzed by Hwang *et al.* [18]. A game-theoretic framework for modeling energy-security tradeoffs due to intrusion detection in wireless sensor networks was proposed by Futaci *et*

al. [11]. A reconfigurable architecture for security using Field Programmable Gate Arrays (FPGAs) was proposed by Gogniat *et al.* [14]. An optimization framework that maximizes security subject to power and delay constraints were respectively proposed by Chandramouli *et al.* [6] and Massey *et al.* [24].

The importance of context for security in mobile environments have been considered by numerous authors [1, 4, 10, 15, 19, 29]. These approaches have proposed adaptable mechanisms which configure security according to the needs of the environment. In contrast, in this thesis context-adaptive energy-aware security is modeled as an optimization problem. Algorithms and a fully polynomial time approximation scheme (FPTAS) for optimum assignment of security levels have been developed towards meeting a given power budget in this thesis.

CONTEXT-SEC has not been explored in the literature. Open problems include determination of its computational complexity and design, analysis, implementation, and testing new algorithms for CONTEXT-SEC. The primary objectives in this research included determining the computational complexity of CONTEXT-SEC, and designing new algorithms for solving the problem depending on its computational complexity status (whether it would be polynomially solvable, or NP-Complete).

CHAPTER 2

NEW ALGORITHMS FOR CONTEXT-SEC

In this chapter, the NP-Completeness of the decision version of CONTEXT-SEC is proved first. Then, three new algorithms for solving the optimization version of CONTEXT-SEC are developed and presented.

2.1 NP-Completeness of CONTEXT-SEC

The exact k item Knapsack Problem (E-kKP), was proved to be NP-Complete in [5]. In this work, E-kKP is reduced to CONTEXT-SEC to prove that CONTEXT-SEC is NP-Complete.

Consider an arbitrary instance of E-kKP with j items ($j \geq k$), where each item i has a weight w_i and a value v_i. The capacity of the knapsack is W. The decision version of this instance of E-kKP can be described as follows: Given an integer V, can exactly k items be put into the knapsack such that $\sum_{a=1}^{k} w_a \leq W$ and $\sum_{a=1}^{k} v_a \geq V$?

Now consider an instance of CONTEXT-SEC with n locations (where the value of n is known, i.e., the number of locations is known) and $m \geq n$ security levels. Each security level i has a power consumption p_i and a user benefit b_i. The energy budget of the user is E. The decision version of this instance of CONTEXT-SEC

can be described as follows: Given an integer B, can exactly one security level be allocated to each of the n locations such that $\sum_{a=1}^{n} p_a \leq E$ and $\sum_{a=1}^{n} b_a \geq B$?

In the reduction, for the instance of CONTEXT-SEC, $n = k$ and $m = j$ are set, and for each level i, $p_i = w_i$, and $b_i = v_i$. The energy budget $E = W$, and the integer $B = V$.

Lemma 1. CONTEXT-SEC is solvable if and only if E-kKP is solvable.

Proof. Assume there is a solution to the arbitrary instance of E-kKP. With the above reduction, there is also a solution to the fixed instance of CONTEXT-SEC with the given allocations. □

On the other side, if there is a solution to the fixed instance of CONTEXT-SEC with the given allocations there is a solution to the arbitrary instance of E-kKP.

Lemma 1 leads to the following theorem.

Theorem 1. CONTEXT-SEC *is* NP-*Complete.*

2.2 New Algorithms for solving CONTEXT-SEC

In this section, the three algorithms for CONTEXT-SEC that were designed in this thesis are described.

2.2.1 An Optimal Dynamic Programming Algorithm

Dynamic programming is a mathematical optimization method that solves problems by combining the solutions to sub-problems. In dynamic programming a complex problem is broken down into simpler, sub-problems, and their solutions are combined to form the final solution to the original problem [8].

A dynamic programming algorithm was designed for E-kKP problem in [22]. Being a similar problem, CONTEXT-SEC can also be solved using a dynamic programming algorithm which computes the optimal assignment of security levels that will maximize user utility and stay within the constraints of a power budget. The dynamic programming algorithm for computing the optimal power consumption meeting the energy budget is described in (Algorithm 1).

Algorithm 1 Dynamic Programming Algorithm

procedure DYNAMIC PROGRAMMING(S, E, n)
  for `i = 1:E` **do**
    for `j = 1:m` **do**
      for `k = 1:n` **do**
        if $j \leq k$ **then**
          $A[i, j, k] \leftarrow \max(A[i, j-1, k],$
            $A[i - p_j, j, k] + b_j)$
        else
          $A[i, j, k] \leftarrow \max(A[i, j-1, k],$
            $A[i - p_j, j, k-1] + b_j)$
        end if
      end for
    end for
  end for
  Output the allocation obtained.
end procedure

The recurrence relations for the dynamic programming algorithm for E-kKP have been used to design the dynamic programming algorithm for CONTEXT-SEC (Algorithm 1) in this work. The following two cases were considered while constructing the recurrence relations for the dynamic programming algorithm (Algorithm 1):

1. For a number of security levels less than or equal to the number of locations to be allocated levels ($j \leq k$), then: $A[i,j,k] = \max\{A[i,j-1,k], A[i-p_j,j,k]+b_j\}$.

2. For a number of security levels greater than the number of locations to be allocated levels ($j > k$), then: $A[i,j,k] = \max\{A[i,j-1,k], A[i-p_j,j,k-1]+b_j\}$.

It would not be necessary to replace any of the k levels allocated to the k locations because it may be that not all of the k locations are allocated a security level (since $j \leq k$) . The second expression works since the $(k-1)$-th layer of the table A keeps track of the best combination of $k-1$ security levels (for $k-1$ locations) among the first $j-1$ security levels as required above. Algorithm 1 lists the dynamic programming algorithm. The worst case run-time complexity of Algorithm 1 can be observed to be $O(Emn)$.

In dynamic programming algorithm (Algorithm 1), an array $A[1\cdots E, 1\cdots m, 1\cdots n]$ is constructed. At the end of executing the dynamic programming algorithm (Algorithm 1) for input array A, any element $A[i,j,k]$ of A (for $1 \leq i \leq E$, $1 \leq j \leq m$ and $1 \leq k \leq n$), contains the maximum user benefit for energy budget i, the first j security levels, and the first k locations. Now, as the nested system of loops in dynamic programming algorithm (Algorithm 1) executes, we consider what needs

to be done for a specific triple (i, j, k). Per sequencing of the i, j, k-loops, when (i, j, k) is reached, the maximum benefit corresponding to the energy budget of i by considering the first j security levels for the first $(k-1)$ locations has been determined. Also $A[i, j-1, k]$ would be the current maximum benefit with the first k locations, considering security level allocations of up to $(j-1)$. Next, to determine the maximum benefit for the energy budget i with the first j levels and the first k locations, there are two cases to consider: one corresponding to $j \leq k$, and one corresponding to $j > k$. If $j \leq k$, the maximum benefit achieved if location k were allocated security level j could be at most b_j in excess of the total benefit achieved when the energy budget was at level $i - p_j$, or $A[i-p, j, j, k] + b_j$. This is why we set $A[i, j, k]$ to $\max(A[i-p_j, j, k] + b_j, A[i, j-1, k])$. In a similar manner, if $j > k$, security level j would have been allocated to some location already. Thus, allocating security level j to location k would increase the benefit at most to $A[i-p_j, j, k-1]+b_j$. Thus, we set $A[i, j, k]$ to $\max(A[i-p_j, j, k-1]+b_j, A[i, j-1, k])$.

Once all of the elements of this array are computed, $A[E, m, n]$ delivers the maximum user benefit (B) and satisfies the constraint that the total power consumption is less than the energy budget E. In other words, $A[E, m, n]$ contains an optimal solution for CONTEXT-SEC. Each security level j is a pair (p_j, b_j), denoting respectively the power consumption and user benefit for level j.

Approach	*Time Complexity*
Brute-force	$O(m^n)$
Dynamic Programming	$O(Emn)$

Table 2.1: Comparison of Runtime Complexity

For a brute force approach, assume $m > n$, with m being the number of security levels, and n being the number of locations. A brute force approach for solving CONTEXT-SEC would be to try out all possible m^n allocations to find out which one yields the best possible total benefit satisfying the energy budget constraint E. Table 2.1 compares the complexities of the brute force and dynamic programming approach.

Ideally, the value of the energy budget E, would be a much higher than the number of locations n, or the number of security levels m (i.e., $E \gg m$, and $E \gg n$). Therefore, if the computational complexity of any algorithm for CONTEXT-SEC depends on (E, m, and n), its computational complexity will predominantly depend on the size of E, or the number of bits z needed to represent E. The run-time for dynamic programming algorithm (Algorithm 1), which is ($O(Emn) \approx O(E)$), grows exponentially in the number of bits z needed to represent E (i.e. is $O(2^z)$).

2.2.2 A Greedy Algorithm

A greedy algorithm for an optimization problem constructs a solution by picking the choice that looks best at the moment and adding it to the current subsolution. Examples of greedy algorithms for well known optimization problems include Dijkstra's shortest path algorithm and Prim/Kruskal's Minimal Spanning Tree algorithms.Greedy algorithms do not always yield optimal solutions but, when they do, they are usually the simplest and most efficient algorithms available [9]. In this work, a greedy algorithm for CONTEXT-SEC has therefore been designed, implemented, and tested.

The greedy algorithm maximizes security appropriate to user context as indicated by user benefits while meeting the given energy budget. The Max-Value heuristic was adapted to this problem. Greedy algorithm (Algorithm 2) contains the description of the Max-Value Greedy Algorithm. Again, S is the set of security levels ($|S| = m$).

Algorithm 2 Greedy Algorithm

```
1:  procedure GREEDY(S, E, n)
2:      e ← 0                                  ▷ e ← temporary energy consumption
3:      b ← 0                                  ▷ b ← temporary benefit
4:      Sort(S, DESCENDING, b_i/p_i).
5:      for i = 1:n do
6:          for j = m-i:m do
7:              if e + p_j ≤ E then
8:                  i ← S_j                    ▷ S_j ← jth security level
9:                  e ← e + p_j
10:                 b ← b + b_j
11:                 break
12:             end if
13:             exit
14:         end for
15:     end for
16:     Output the allocation obtained.
17: end procedure
```

Step 1 of greedy algorithm (Algorithm 2) has an execution time of $O(m \log m)$, due to the sorting step for m levels. Step 2 has a complexity of $O(n)$ due to a maximum allocation of n levels to the n locations. Step 4 has a single check, which accounts for a constant time. Step 5 is also a single output accounting for a constant time. Ideally, $m \log m > n$, therefore, the computational complexity of Algorithm 2 is $O(m \log m)$.

Note that the final allocation output of greedy algorithm (Algorithm 2) might not have a security level allocated for every location, since there might not exist any valid allocation of levels to locations satisfying the energy budget constraint E.

2.2.3 A Fully Polynomial Time Approximation Scheme (FPTAS)

The idea behind the FPTAS algorithm is to scale down the value of the energy budget E by a logarithmic factor, such that the scaled-down budget $\bar{E} \approx \log E$. If dynamic programming algorithm (Algorithm 1) is now executed with $\bar{E}$ as input, the resulting computational complexity of the algorithm becomes $O(\bar{E}) = O(\log E) = O(\log 2^z) = O(z)$. But to ensure that allocation output with the scaled-down energy budget $\bar{E}$ is not much worse than that with the original budget E, the power consumption of each level ($p_i's$) has to be also scaled down by the same factor as E. This idea has been used to obtain an FPTAS for CONTEXT-SEC. In particular, the power consumption and energy budget can be scaled down enough so that they are polynomially bounded in m (the number of levels), using dynamic programming (Algorithm 1) in the new instance, and will then return the resulting solution (allocation). By rounding with respect to the desired ε/m, a solution can be found that is at least $(1 - \varepsilon) \times OPT$ in polynomial time with respect to both m and $1/\varepsilon$, giving an FPTAS where $1 \geq \varepsilon > 0$, as illustrated in Algorithm 3.

Algorithm 3 FPTAS

```
1: procedure FPTAS(S, E, n)
2:     ε ∈R_{≥0}                          ▷ Choose ε, such that 1 ≥ ε ≥ 0
3:     Ē ← ⌈εE/m⌉.
4:     for i = 1:m do
5:         p̄_i ← ⌈εp_i/m⌉
6:         S̄ ← (p̄_i, b_i)
7:     end for
8:     DYNAMIC PROGRAMMING(S̄, Ē, n).
9:     Output the allocation.
10: end procedure
```

As before, S is the set of security levels with $|S| = m$. The chosen ε value should preserve the ratio of the maximum power consumption value across security levels to the energy budget. In other words, ε should be chosen such that $E/\max(p_i) \approx \bar{E}/\max(\bar{p}_i)$ to assure that the solution produced by FPTAS algorithm (Algorithm 3) is not much worse than the dynamic programming algorithm (Algorithm 1).

2.3 Comparison of Computational Complexities

Algorithm	*Computational Complexity*
Dynamic Programming	$O(E \times m \times n)$
FPTAS	$O(\log E \times m \times n)$
Greedy	$O(m \times \log m)$

Table 2.2: Comparison of the three algorithms.

Table 2.2 compares the three algorithms based on their computational complexities. The complexity of the greedy algorithm (Algorithm 2) does not depend on the value of the energy budget E as the other two algorithms. This is supported later by the experimental results presented in Figure 4.2, where the greedy algorithm exhibits much lesser execution time compared to the other two algorithms over the same input.

CHAPTER 3

IMPLEMENTATION OF THE ALGORITHMS

This chapter contains implementation details of the three algorithms designed. Additionally, examples of execution of the designed algorithms over a given input have been illustrated in this chapter.

3.1 Implementing the Dynamic Programming Algorithm

The dynamic programming algorithm (Algorithm 1) builds up a three-dimensional array. The last value of the array gives the final solution, as explained in Section 2.2.

In the code, which was implemented using `C++` language, a three-dimensional array was used, and the recurrences (refer to Section 2.2) were implemented to fill in the values of the array. An initial execution of the dynamic programming algorithm over a given input is described below.

For example, if the number of security levels (m)= 5, the pairs of security benefit and energy consumption (p_i, b_i) for each level i will be: $\{(12, 24), (7, 13), (11, 23), (8, 15), (9, 16)\}$ for Energy Budget (E)= 26 and the Number of locations (n) = 3

For the above input, dynamic programming algorithm (Algorithm 1) generates a $26 \times 5 \times 3$ table, where each cell $A[i, j, k]$ has the optimal solution (total benefit value), with $E = i$ (energy budget of i), $m = j$ (the first j security levels), and $n = k$, (k locations). Once dynamic programming algorithm (Algorithm 1) has executed and computed values of all $A[i, j, k]$, the last cell $A[26, 5, 3]$ (assuming indexes start from $A[0, 0, 0]$) contains the optimal solution for the problem. It is easy to find the allocations that result in this optimal solution using backtracking once the array A is computed by dynamic programming algorithm (Algorithm 1). But in this work, the benefit value of the solution (value of $A[E, m, n]$) is of primary interest. This three-dimensional table can be projected in two dimensions by having 26 tables of dimensions 5×3. The 26^{th} table in this case has the final solution (the total benefit value).

Each of Tables 3.1, 3.2, and 3.3, have to be interpreted in the following way:

- Index i represents the value of energy budget E (as mentioned above).
- Rows represent the number of security levels m ranging from 1 through 5.
- Columns represent the number of locations n ranging from 1 through 3.
- Each cell (a, b) of a table contains the total benefit value for $E = i$, $m = a$, and $n = b$ as output by the dynamic programming algorithm (Algorithm 1).
- As an example cell $(3, 2)$ of Table 3.3 contains the maximum benefit value (47 in this case), when there are three levels (in this case the levels are $\{(12, 24), (7, 13), (11, 23)\}$), two locations to allocate levels, and energy budget $E = 26$.

	n_1	n_2	n_3
m_1	0	0	0
m_2	0	0	0
m_3	0	0	0
m_4	0	0	0
m_5	0	0	0

Table 3.1: Table for $i = 1$

Table 3.1 gives the solution for $E = 1$ (energy budget), $m = 5$ (number of security levels), $n = 3$ (number of locations). Since $p_j \geq 7, \forall j$, the seventh table (table for $i = 7$) has a solution greater than 0. So Table 3.1 gives a correct benefit solution of 0.

	n_1	n_2	n_3
m_1	0	0	0
m_2	13	13	13
m_3	13	13	13
m_4	13	13	13
m_5	13	13	13

Table 3.2: Table for $i = 7$

Table 3.2 gives the solution for $E = 7$ (energy budget), $m = 5$ (number of security levels), $n = 3$ (number of locations). Using the dynamic programming algorithm (Algorithm 1), allocating the second security level to any of the locations meets the energy budget of 7, and generates a benefit of 13.

Figure 3.1 (derived from Table 3.3) gives the final solution for $E = 26$ (total power budget), $m = 5$ (number of security levels), $n = 3$ (number of locations). The proof of optimality of this solution has not been formally described in this thesis,

	n_1	n_2	n_3
m_1	24	24	24
m_2	24	37	37
m_3	24	47	47
m_4	24	47	51
m_5	24	47	51

Table 3.3: Table for $i = 26$

though it can be claimed to be optimal since 1) the principle of optimality holds for CONTEXT-SEC, because KNAPSACK was reduced to it [16, 26], and 2) the dynamic programming algorithm (Algorithm 1) arrives at this solution by considering all possible values for i, j, and k.

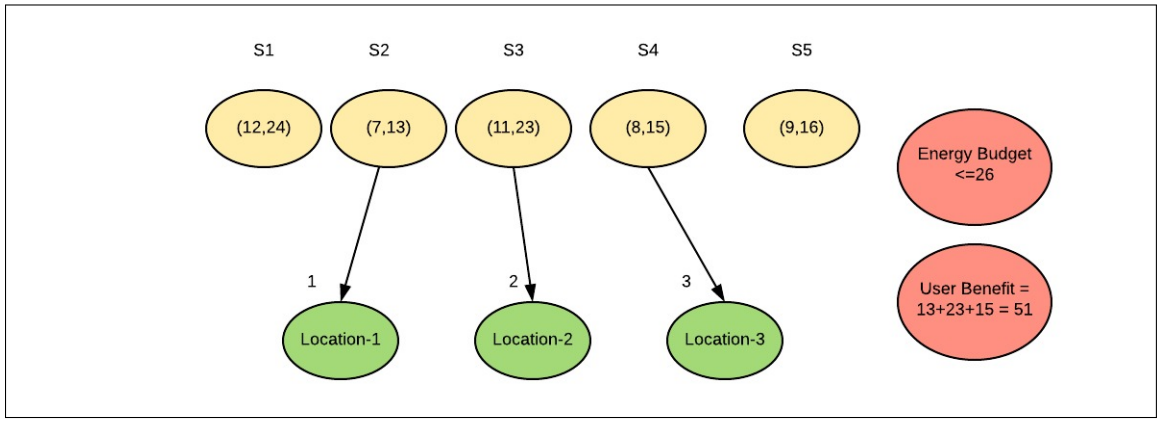

Figure 3.1: Final solution of Algorithm 1

Though the actual allocation of levels to the locations is not of prime importance for CONTEXT-SEC (since we are only comparing the total benefit output for different algorithms), the actual allocation generated by the dynamic programming Algorithm (Algorithm 1) and FPTAS Algorithm (Algorithm 3) can be obtained by adding the same backtracking logic as described for the dynamic programming algorithm (Algorithm 1) for KNAPSACK [8, 21]. The logic is summarized here. Once the three dimensional array A is constructed by the dynamic programming algorithm (Algorithm 1), a security level j is included in the solution

if $A[i,j,k] \neq A[i,j-1,k]$. The final solution will therefore contain j levels where $j \leq n$.

Following the above logic, the final solution of the dynamic programming algorithm (Algorithm 1) for the given input is the allocation of security levels $(7,13)$, $(11,23)$, and $(8,15)$ to the three locations for a total user benefit of 51, which is optimal. It is to be noted that the current version of CONTEXT-SEC does not differentiate between locations; in other words, since locations do not have security preferences, the three levels ($(7,13)$, $(11,23)$, and $(8,15)$) output by dynamic programming Algorithm, could be assumed to be allocated to the three locations in any order.

3.2 Implementing the Greedy Algorithm

The greedy algorithm (Algorithm 2) sorts the array of security levels in descending order of $\frac{b_i}{p_i}$ and then greedily selects levels from the sorted array until the energy budget constraint is no longer satisfied. The execution of the greedy algorithm (Algorithm 2) is demonstrated on the same input. Let us consider the number of security levels $(m) = 5$, the pairs of security benefit and energy consumption (p_i, b_i) for each level $i = \{(12,24),(7,13),(11,23),(8,15),(9,16)\}$ with an energy budget $(E) = 26$ and number of locations $(n) = 3$. The first step of the greedy algorithm (Algorithm 2) is to sort the security levels based on $\frac{b_i}{p_i}$, $\forall i$. For the first level, $(12,24)$, $\frac{b_1}{p_1} = \frac{24}{12} = 2$, second level, $(7,13)$, $\frac{b_2}{p_2} = \frac{13}{7} = 1.8571$, third level, $(11,23)$, $\frac{b_3}{p_3} = \frac{23}{11} = 2.09$, fourth level, $(8,15)$, $\frac{b_4}{p_4} = \frac{15}{8} = 1.8750$, fifth level, $(9,16)$, $\frac{b_5}{p_5} = \frac{16}{9} = 1.7777$. Hence the resultant sorted security levels based on $\frac{b_i}{p_i}$ will be $\{(11,23),(12,24),(8,15),(7,13),(9,16)\}$.

The algorithm now greedily selects levels until the energy constraint $E = 26$ is satisfied. The final solution obtained by of the greedy algorithm (Algorithm 2) on the given input is a total user benefit of 47, which is achieved by allocating security levels $(11, 23)$ and $(12, 24)$ to the first two locations, while the third location is not allocated a security level.

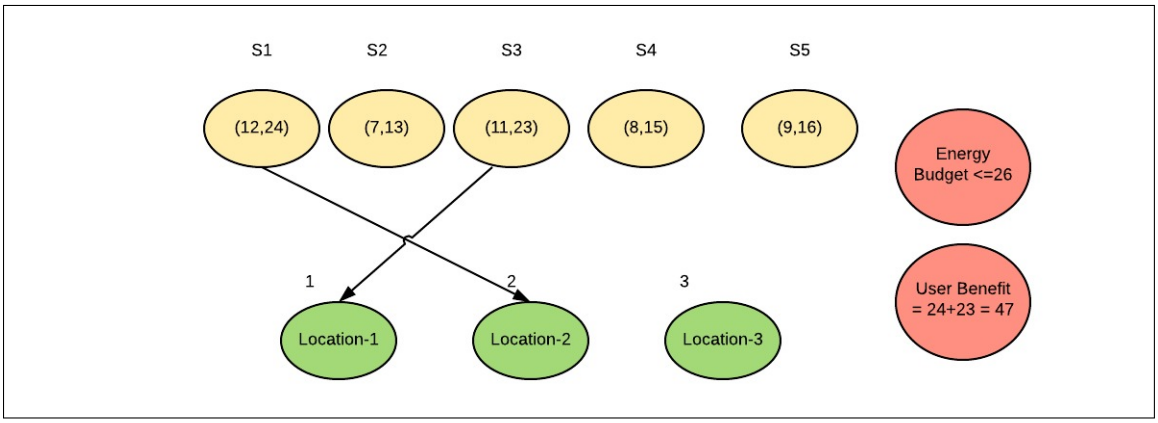

Figure 3.2: Final solution of Algorithm 2

Figure 3.2 shows the final solution allocation to be $\{S_3, S_1, -\}$. These levels are allocated with a total energy consumption of 23, and a total benefit of 47. Location 3 is not allocated a level by the greedy algorithm (Algorithm 2). Comparing the total benefit (of 47) with that produced by the dynamic algorithm (Algorithm 1) (51) tells us this is not the optimal solution.

3.3 Implementing the FPTAS Algorithm

The FPTAS algorithm (Algorithm 3) has a function to scale down the energy budget E and the power consumption p_i for each level i. The dynamic Algorithm (Algorithm 1) is then executed on the scaled-down input. It therefore produces an

array $\bar{A}$ of dimensions $\bar{E} \times m \times n$. The following is the result of executing thew FPTAS on the same example input (as above):

The number of security levels $(m) = 5$, pairs of security benefit and energy consumption (p_i, b_i) for each level $i = \{(12, 24), (7, 13), (11, 23), (8, 15), (9, 16)\}$ with energy Budget $(E) = 26$ and number of locations $(n) = 3$.

For the above small example, choose $\varepsilon = 1$. $\bar{E} = \left\lceil \frac{\varepsilon E}{m} \right\rceil = \frac{26}{5} = 6$, $\bar{p_1} = \left\lceil \frac{\varepsilon p_1}{m} \right\rceil = \frac{12}{5} = 3$, $\bar{p_2} = \left\lceil \frac{\varepsilon p_2}{m} \right\rceil = \frac{7}{5} = 2$, $\bar{p_3} = \left\lceil \frac{\varepsilon p_3}{m} \right\rceil = \frac{11}{5} = 3$, $\bar{p_4} = \left\lceil \frac{\varepsilon p_4}{m} \right\rceil = \frac{8}{5} = 2$ and $\bar{p_5} = \left\lceil \frac{\varepsilon p_5}{m} \right\rceil = \frac{9}{5} = 2$.

Therefore the scaled-down input for FPTAS becomes:- number of security levels (m): 5, the pairs of security benefit and energy consumption (p_i, b_i) for each level i: $\{(3, 24), (2, 13), (3, 23), (2, 15), (2, 16)\}$ with an energy budget (E)= 6 and number of locations (n)= 3

For the above input, the dynamic algorithm (Algorithm 1) (a sub-algorithm of FPTAS algorithm (Algorithm 3) generates a $6 \times 5 \times 3$ array $\bar{A}$, as opposed to $26 \times 5 \times 3$ array A produced by the dynamic algorithm (Algorithm 1) originally on this input. The final output from FPTAS algorithm (Algorithm 3) is $\{S_1, S_3, -\}$ (Figure 3.3), the set of the two security levels $(12, 24)$ and $(11, 23)$ for the first two locations. The third location is not allocated a security level. Figure 3.3 depicts this solution.

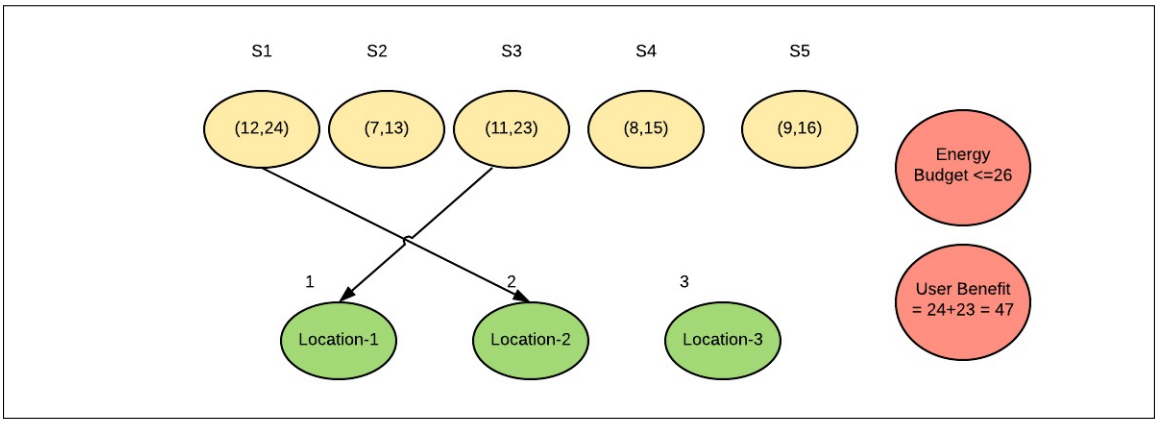

Figure 3.3: Final solution of Algorithm 3

3.4 Data Structures Used

A three-dimensional array is used in the dynamic programming algorithm (Algorithm 1). It was initialized with zeros and the values were populated later using dynamic programming.

A simple one-dimensional array was used in the greedy algorithm (Algorithm 2). The pairs of security benefit and power consumption were elements of the array and they were sorted based on the ratio of security benefit and power consumption of each element.

The FPTAS algorithm (Algorithm 3) runs the dynamic programming algorithm (Algorithm 1) on a scaled-down input, and therefore also uses a three dimensional array A for its processing. A separate function has been implemented to scale down the energy budget and the power consumption of each level.

CHAPTER 4

RESULTS AND DISCUSSION

Two different data-sets were used to compare the three algorithms. The values of the solution outputs from the three algorithms and the execution time to produce their respective solutions are compared in this chapter.

4.1 Data-Sets

Each algorithm was executed over two data-sets derived from the Live-lab project [30]. Live-Lab is a methodology to log smart-phone users in the field and to analyze the wireless networks used by those smart-phone users. The key features of Live-Lab include:

1. Comprehensive in-device logging of smart-phone usage and measurement of activity of wireless networks.

2. In-field programmability of the logger so that researchers can update the logger and schedule a new measurement very much like they would do with a lab computer.

3. A large number of users who use the logged smart-phones as their primary phones for a long term (one year).

Parameter	Value Data-Set-1	Value Data-Set-2
# locations (n)	20	17
# security levels (m)	3380	289
Energy budget (E)	1700	1445

Table 4.1: Data-Sets

Two data-sets were derived from the Live-Lab project with the parameters shown in Table 4.1. Three tables `power_detail.sql, associatedwifi.sql, cellsignal .sql` have been used to derive the data-sets for the implementation work of this thesis. The data have been collected for a single mobile user (A00) for `uid` column common to all three tables. Data was extracted from all three tables for the mobile user (A00), for a particular range of the time (of six months) from (08/23/2010) to (02/24/2011) which produced huge lists of records. Two data-sets were created from these records; one spanning the first two months (resulting in a smaller data-set) and the other spanning over the last four months resulting in a larger data-set. The number of locations n for each data-set was obtained by the number of unique `bssids` in the data-set. BSSID is the MAC address of the wireless access point (WAP) generated by combining the 24 bit Organization Unique Identifier (the manufacturer's identity) and the manufacturer's assigned a 24-bit identifier for the radio chip-set in the WAP. The `bssid` column in the data listed the `bssids`. Next each of the m security levels i is again an ordered pair (p_i, b_i), where p_i and b_i are respectively the power consumption and security benefit for that level. The power consumption p_i for each level i has been derived from the battery voltage `mah` field with unit `millivolts`, and the benefit b_i for each level i has been computed by averaging the cell signal quality `csq` and bit error rate `ber` respectively from the data-sets [30]. The energy budget was set arbitrarily based on the total power consumption across levels, after performing a few rounds of experimental execution. The author of this thesis will share the two data-sets derived, with any researchers

interested in reproducing the experiments or enhancing this research work.

4.2 Results

Figure 4.1 shows the solution values for user benefit produced by the three algorithms over the two data-sets. As seen in Figure 4.1, the greedy algorithm (Algorithm 2) produced a value approximately half the optimal value produced by the dynamic programming algorithm (Algorithm 1). Interestingly, for both the data-sets, the solution value produced by the FPTAS algorithm(Algorithm 3) was very close to the optimal value produced by dynamic programming algorithm (Algorithm 1). As a matter of fact, for data-set-2, FPTAS algorithm (Algorithm 3) produced the optimal solution.

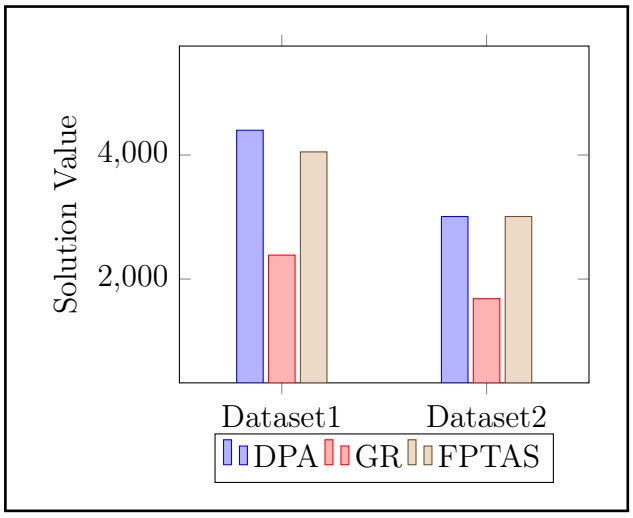

Figure 4.1: Solution values produced for the two data-sets.

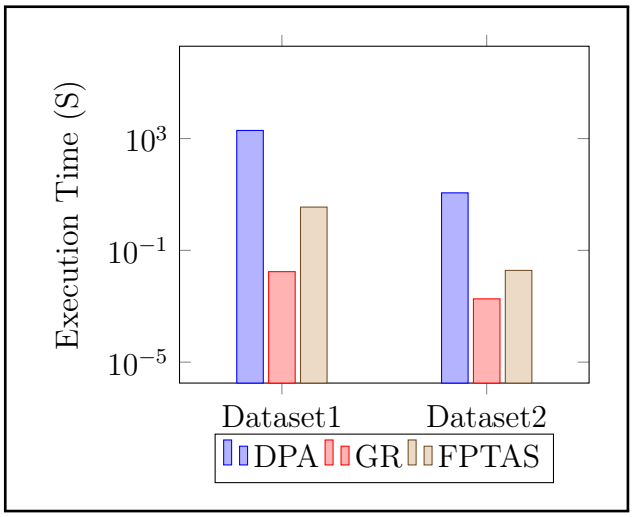

Figure 4.2: Execution Time in seconds.

Figure 4.2 illustrates the time taken by each algorithm to execute over the two data-sets. For clarity of comparison, a log plot is used in the figure. As expected, the dynamic programming algorithm (Algorithm 1) took the longest time (since it is an exponential algorithm). The greedy algorithm (Algorithm 2) had the shortest execution time. The FPTAS algorithm (Algorithm 3) did better than

dynamic programming algorithm (Algorithm 1), though it was not able to match the greedy algorithm (Algorithm 2).

4.3 Discussion

Based on the plot of Figure 4.1 the dynamic programming algorithm (Algorithm 3.1) proves to be the algorithm that produces the best results in terms of total benefit. Based on the earlier analysis, the dynamic programming algorithm (Algorithm 1) is an optimal algorithm for CONTEXT-SEC. The greedy and FPTAS algorithms were expected to approximate the optimal solution well; they execute in polynomial time. FPTAS outperformed greedy in terms of solution value, but it took considerably more execution time. The execution time of FPTAS depends on the ε value chosen, and the best value again might depend on the input.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this work, a context-adaptive energy-aware approach to security was developed for mobile devices. In particular, the problem of context-adaptive energy-aware security was modeled as a combinatorial optimization problem (CONTEXT-SEC), and three different algorithms were designed to solve it.

Additionally, it was proven that the decision version of the problem is NP-Complete through a reduction from KNAPSACK. The three algorithms were then implemented and their performance were measured on real-world datasets.

5.2 Future Work

The next steps would be to add the user preference constraint to each location. In other words, each location would have a preferred security benefit. The goals would remain the same but with the additional constraint to satisfy as many locations as possible with their preferred levels. Also, an online version of the problem, where the locations and/or security levels are not available at the outset, will need to be explored.

REFERENCES

[1] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. Towards a better understanding of context and context-awareness. In Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (London, UK, UK, 1999), HUC '99, Springer-Verlag, pp. 304–307.

[2] Al-Turjman, F., and Alturjman, S. Context-sensitive access in industrial internet of things (iiot) healthcare applications. IEEE Transactions on Industrial Informatics 14, 6 (2018), 2736–2744.

[3] Ben-Asher, N., Kirschnick, N., Sieger, H., Meyer, J., Ben-Oved, A., and Möller, S. On the need for different security methods on mobile phones. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (2011), ACM, pp. 465–473.

[4] Botezatu, N., Manta, V., and Stan, A. Self-adaptability in secure embedded systems: an energy-performance trade-off. In The 2011 International Conference on Information Security and Internet Engineering, World Congress on Engineering (2011).

[5] Caprara, A., Kellerer, H., Pferschy, U., and Pisinger, D. Approximation algorithms for knapsack problems with cardinality constraints. European Journal of Operational Research 123, 2 (2000), 333–345.

[6] Chandramouli, R., Bapatla, S., Subbalakshmi, K., and Uma, R. Battery power-aware encryption. ACM Transactions on Information and System Security (TISSEC) 9, 2 (2006), 162–180.

[7] Chang, C.-C., Nagel, D., and Muftic, S. Balancing security and energy consumption in wireless sensor networks. Mobile Ad-Hoc and Sensor Networks (2007), 469–480.

[8] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Dynamic programming. Introduction to Algorithms, (2001), 323–370.

[9] Dekai, W. U. Lecture 14: Greedy algorithms (clrs section 16). `https://home.cse.ust.hk/~dekai/271/notes/L14/L14.pdf`. Accessed: 2019-04-11.

[10] Fischer, G. Context-aware systems: the 'right' information, at the 'right' time, in the 'right' place, in the 'right' way, to the 'right' person. In Proceedings of the International Working Conference on Advanced Visual Interfaces (New York, NY, USA, 2012), AVI '12, ACM, pp. 287–294.

[11] Futaci, S. M., Jaffrès-Runser, K., and Comaniciu, C. On modeling energy-security trade-offs for distributed monitoring in wireless ad hoc networks. In Military Communications Conference, 2008. MILCOM 2008. IEEE (2008), IEEE, pp. 1–7.

[12] Ganesan, P., Venugopalan, R., Peddabachagari, P., Dean, A., Mueller, F., and Sichitiu, M. Analyzing and modeling encryption overhead for sensor network nodes. In Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (2003), ACM, pp. 151–159.

[13] Ghahramani, M., Zhou, M., and Hon, C. T. Mobile phone data analysis: A spatial exploration toward hotspot detection. IEEE Transactions on Automation Science and Engineering (2018).

[14] Gogniat, G., Wolf, T., and Burleson, W. Reconfigurable security primitive for embedded systems. In System-on-Chip, 2005. Proceedings. 2005 International Symposium on (2005), pp. 23–28.

[15] Hager, C. T. Context aware and adaptive security for wireless networks. PhD thesis, Virginia Polytechnic Institute and State University, 2004.

[16] Helman, P. The principle of optimality in the design of efficient algorithms. Journal of Mathematical Analysis and Applications 119, 1-2 (1986), 97–127.

[17] Hodjat, A., and Verbauwhede, I. The energy cost of secrets in ad-hoc networks (short paper), 2002.

[18] Hwang, D. D., cheng Charles Lai, B., and Verbauwhede, I. Energy-memory-security tradeoffs in distributed sensor networks. In In ADHOC-NOW (2004), pp. 70–81.

[19] Jameson, A. Modelling both the context and the user. Personal Ubiquitous Comput. 5, 1 (Jan. 2001), 29–33.

[20] Karri, R., and Mishra, P. Optimizing ipsec for energy-efficient secure wireless sessions. In System-Level Power Optimization for Wireless Multimedia Communication. Springer, 2002, pp. 133–152.

[21] Kleinberg, J., and Tardos, E. Algorithm design. Pearson Education India, 2006.

[22] López, C. L. Variant of the knapsack problem. https://cs.stackexchange.com/questions/18492/variant-of-the-knapsack-problem. Accessed: 2018-01-11.

[23] Lee, J., Kapitanova, K., and Son, S. H. The price of security in wireless sensor networks. Comput. Netw. 54, 17 (Dec. 2010), 2967–2978.

[24] Massey, T., Brisk, P., Dabiri, F., and Sarrafzadeh, M. Delay aware, reconfigurable security for embedded systems. In Proceedings of the ICST 2nd international conference on Body area networks (ICST, Brussels, Belgium, Belgium, 2007), BodyNets '07, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 12:1–12:5.

[25] McKay, K. Trade-offs Between Energy and Security in Wireless Networks. PhD thesis, Worcester Polytechnic Institute, 2005.

[26] Posypkin, M., and Sin, S. T. T. Comparative analysis of the efficiency of various dynamic programming algorithms for the knapsack problem. In 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW) (2016), IEEE, pp. 313–316.

[27] Potlapally, N. R., Ravi, S., Raghunathan, A., and Jha, N. K. Analyzing the energy consumption of security protocols. In Proceedings of the 2003 international symposium on Low power electronics and design (2003), ACM, pp. 30–35.

[28] Sankaran, S. Lightweight security framework for iots using identity based cryptography. In 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (Sept 2016), pp. 880–886.

[29] Sankaran, S., and Sridhar, R. User-adaptive energy-aware security for mobile devices. In Communications and Network Security (CNS), 2013 IEEE Conference on (2013), IEEE, pp. 391–392.

[30] Shepard, C., Rahmati, A., Tossell, C., Zhong, L., and Kortum, P. Livelab: measuring wireless networks and smartphone users in the field. ACM SIGMETRICS Performance Evaluation Review 38, 3 (2011), 15–20.

[31] Zhu, K. Phone usage pattern as credit card fraud detection trigger, Feb. 26 2013. US Patent 8,386,386.

LIST OF PUBLICATIONS FROM THE THESIS

1. Roy, S., Sankaran, S., Singh, P., & Sridhar, R. Modeling Context-Adaptive Energy-Aware Security in Mobile Devices. The 43nd IEEE Conference on Local Computer Networks (LCN), 2018 (Accepted, to appear).

2. Roy, S., Sankaran, S., Singh, P., Sridhar, R., & Asaithambi, A. Context-sec: Balancing Energy Consumption and Security of Mobile Devices. The 9th International Green and Sustainable Computing Conference (IGSC), 2018 (Accepted, to appear).

VITA

Preeti Singh received a Bachelor's degree in Computer Science from MDU University, Haryana in India in 2000, she also completed a Master's degree in Computer Applications from IGNOU, New Delhi, India, in 2005. She looks forward to receiving her Master's of Science in Computer and Information Sciences from The University of North Florida in April 2019. Dr. Swapnoneel Roy of School of Computing, University of The North Florida, is Preeti's thesis advisor. Preeti worked as a Graduate Assistant on campus while doing her studies. She also has vast experience of 16+ years working with various software companies as Sr. Software Engineer and IT Manager. She currently works as an IT Manager for Cognizant Technology Solutions and Sr. Software Developer with PNC Bank. Her expertise is in Software Application development and Project Management. Preeti's long term goal is to have her own software company, develop secure applications, and educate students leveraging her experience and development work.