Abstract

Minimizing power consumption is crucial in battery power limited secure wireless mobile networks. In this paper, we (a) introduce a hardware/software set-up to measure the battery power consumption of encryption algorithms through real-life experimentation, (b) based on the profiled data propose mathematical models to capture the relationships between power consumption and security, and (c) formulate and solve security maximization subject to power constraints. Numerical results are presented to illustrate the gains that can be achieved in using solutions of the proposed security maximization problems subject to power constraints.

Battery Power-aware Encryption

R. Chandramouli, S. Bapatla, K.P. Subbalakshmi
B 315, Department of Electrical and Computer Engg.
Stevens Institute of Technology, NJ
Email: {mouli,ksubbala}@stevens.edu

and

R.N. Uma
Department of Mathematics & Computer Science
North Carolina Central University
Email: rnumanccu.edu

---

1. INTRODUCTION

Mobile wireless network security has emerged as an important topic of research recently. It is well known that security approaches based on encryption play a major role in this domain. One major hurdle in providing information security in mobile wireless devices is the limited battery power. The pace of advancements in battery technologies has not kept up with that of wireless technologies [Buchmann]. This implies that mobile devices typically operate on a frugal power budget and therefore computationally intensive encryption/decryption algorithms and the related security parameters may not be supported. We also note that there are several applications such as wireless sensor networks where the battery power limitation is extreme and re-charging or changing drained batteries may be impossible.

Traditional cryptography (crypto) deals mainly with ensuring the security of the encrypted data. Several notions such as provably secure, unconditionally secure etc. are used towards this objective. One key point to note is that the design of crypto algorithms based on these notions typically do not account for physical constraints such as limited battery power. Therefore, as we will see later in this paper, introduction of battery power constraint changes the encryption based security problem significantly. For example, it may not be possible to support a security requirement given the power budget and conversely some security has to be sacrificed so that the physical wireless connection is not cut-off due to total battery power loss. Then the key questions are two fold:

—How do we model the relationship between battery power consumption and crypto algorithms and the related choice of parameters?

—How to optimize crypto parameters for a given power budget?

Therefore, the primary challenge in providing security in low power mobile wireless devices lies in the conflicting interest between minimizing power consumption and maximizing security. In general, we can safely assume that by doing more compu-
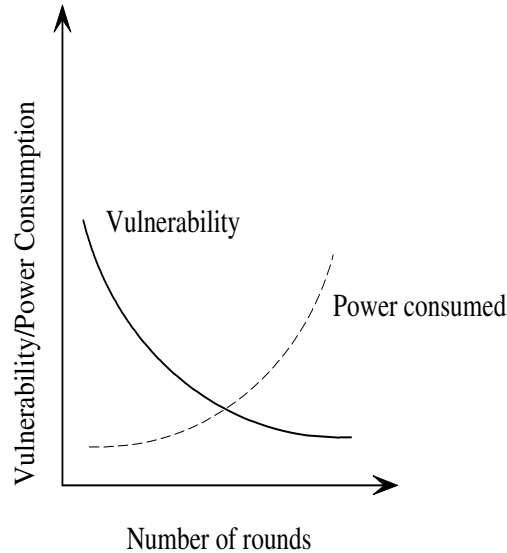
Fig. 1. Security vs. battery power consumption trade-off.

tations one can achieve a higher amount of security. For example, the strength of encryption schemes depend on the size of the key and the number of encryption rounds [Nechvatal et al. 1999]. Larger key sizes/number of rounds produce higher levels of security at the cost of additional power consumption. For example, Fig. 1 shows a general trend in the trade-off between vulnerability and power consumption with varying number of encryption rounds. Therefore, in order to design power efficient encryption algorithms for wireless networks there is an inherent need to understand the relationships between power consumption and encryption parameters. Once these relationships are understood well then it is possible to optimize power consumption w.r.t. a security requirement or vice-versa.

Clearly, there are several approaches to understand and address the limited battery power problem in mobile security. Some of them are the following:

—Power-efficient hardware implementation (e.g., [Ravi et al. 2002; Goodman and Chandrakasan 1998; Sklavos and Koufopavlou 2001; Goodman et al. 1999]).

—Characterization studies of power consumption of different crypto algorithms (e.g., [Potlapally et al. 2003; Hodjat and Verbauwhede 2002; Karri and Mishra 2003; Bapatla and Chandramouli 2004; Prasithsangaree and Krishnamurthy 2003]).

—Lightweight security mechanisms (e.g., [Michell and Srinivasan 2004; Tosun and Feng 2001; Perrig et al. 2002; Slijepcevic et al. 2002]).

Encryption can be implemented in different layers of the network protocol stack, typically, in the link layer or application layer or both. For example, security is

implemented in data link layer in the IEEE 802.11b [IEEE802.11b] wireless local area network standard. At the application layer the content to be encrypted and transmitted can be easily classified into more important (higher security) and less important components (less security). An example is the more important motion vector data and less important quantized AC coefficients in MPEG-4 [MPEG4] video coded bit stream. If such a classification is available then it facilitates the optimization of power consumption via unequal protection of the bit stream subject to an (average) security constraint [Bapatla and Chandramouli 2004]. Conversely, we can pose the problem of security maximization w.r.t. a total power budget as a constrained optimization problem over the different security levels. Encryption can also be turned off or on depending on the time-varying battery power levels.

Two broad classes of cryptographic algorithms are symmetric or secret key cryptography and asymmetric or public key cryptography. In symmetric cryptographic schemes the sender and receiver exchange a secret key before transmission begins. The sender encrypts the message using this secret key and the receiver uses the same key to decrypt it. Typically the computations involved in encryption and decryption using symmetric cryptography are about the same. In public key cryptography the encryption and decryption keys are different. It is known that public key cryptography is highly computationally intensive compared to symmetric cryptography. This is one of the reasons it is not popular for low power secure mobile communications. Also, there is no clear advantage in choosing public key encryption techniques over secret key encryption.

Quantification of the security against attacks provided by an encryption algorithm is difficult. It is still an open research problem. Some popular definitions of security include *computationally security* and *unconditional security*. An encryption scheme is said to be computationally secure if successfully attacking it is as hard as a known "hard" problem such as integer factoring. Here, the computational power of the adversary is assumed to be limited. On the other hand, unconditional security is an information theoretic definition requiring no assumptions on the computational capability of the adversary. As noted in [Nechvatal et al. 1999], the best we can hope for in quantifying security is an estimate based on known cryptanalysis techniques. We follow this philosophy in this paper and provide a definition of vulnerability (or loss in security) based on a brute force cryptanalysis attack. This measure will then be used to investigate the battery power consumption vs. security trade-off.

The goals of this paper are two fold: (a) measuring and modelling power consumption of crypto algorithms and (b) minimize vulnerability subject to a power constraint. We use real-life experimentation based profiling to measure the power consumption of different encryption algorithms. We consider DES, IDEA, GOST and RC4 encryption algorithms. Statistical regression techniques are then applied to the collected data to derive mathematical models that capture the trends in power consumption. This is one of the significant differences between the proposed work and the related work cited in this paper. After developing these models the second goal of this paper is to set-up power efficient encryption as an optimization problem—another major difference compared with the related work in this topic. We define the vulnerability metric as a quantity dependent on the success

probability of a cryptanalysis attack. This metric is used as the objective function to formulate two vulnerability minimization problems. Algorithms that solve these problems to give optimal power and encryption rounds allocation are also presented.

The paper is organized as follows. We present a brief overview of the different block encryption schemes in Section 2, the experimental setup to compute the power profile of encryption algorithms and their mathematical modelling are discussed in Section 3. Section 4 describes the mathematical optimizations of security under power constraint. Numerical results are presented in Section 5. This is followed by concluding remarks in Section 6.

2. OVERVIEW OF ENCRYPTION ALGORITHMS

In this section we provide a brief overview of the block and stream ciphers we have considered in this paper, namely, DES [DES 1977], IDEA [Lai 1992], GOST [GOST 1989] and RC4 [Rivest 1992]. This overview will help to understand the different computational operations and requirements of these methods. Further details can be found in many standard books on encryption such as [Schneier 2002].

2.1 DES

DES is a symmetric block cipher that encrypts data in 64-bit blocks. A 64-bit block of plaintext is taken as input by the algorithm and a 64 bit block ciphertext is produced as output. Since DES is a symmetric cipher the computations performed by the encryption and decryption algorithms are nearly the same. The encryption key length is 56 bits long with an additional 8 bits for parity checking. The algorithm uses two basic components: substitution and permutation. A single application of these two components on the plaintext using the key is called a *round*. The maximum number of rounds in DES is 16.

Before the first round the 64-bit block plaintext is permuted using an initial permutation (IP) table. Then the permuted block is divided into a left and a right half of 32-bits each. Then a set of operations called Function f (Feistel network)is applied in each round that combines the data with the encryption keys. The computations involved in the f-function are as follows. The encryption key for each round is derived by permuting the key bits using a key permutation table and then choosing 48 bits. The right half of the 32 bit data block is expanded to 48 bits using an expansion box, XORed with the key and then sent through 8 substitution boxes (S-box) to produce 32 bits. These 32 bits are permuted again.

The output of the f-function is combined with the left half of 32 bits via XOR. This is then swapped with the right half of data. This could be repeated up to a maximum of 16 rounds. Decryption is the exact inverse operation.

2.2 IDEA

IDEA operates on 64 bit plaintext blocks with 128 bit key. This is also a symmetric encryption technique and the same algorithm is used for both encryption and decryption. The operations on 16-bit sub-blocks are XOR, addition modulo 2^{16} and multiplication modulo $2^{16} + 1$.

There are totally 8 rounds in IDEA. Initially the 64 bit block is divided into four 16 bit sub-blocks (say, X_1, X_2, X_2 and X_4) and given as input to the first

round of the algorithm. The four sub-blocks are subjected to the three operations listed above, the second and third sub-blocks are swapped and given as input to the next round. This process continues for 8 rounds where in each round the following computations are performed [Schneier 2002]:

(1) Multiply X_1 and the first subkey
(2) Add X_2 and the first subkey
(3) Add X_3 and the third subkey
(4) Multiply X_4 and the fourth subkey
(5) XOR the results of steps 1 and 4
(6) XOR the results of steps 2 and 4
(7) Multiply the results of steps 5 with the fifth subkey
(8) Add the results of steps 6 and 7
(9) Multiply the results of step 8 with the sixth subkey
(10) Add the results of steps 7 and 9
(11) XOR the results of steps 1 and 9
(12) XOR the results of steps 3 and 9
(13) XOR the results of steps 2 and 10
(14) XOR the results of steps 4 and 10

After a round four sub-blocks are generated by steps 11, 12, 13 and 14. The two inner blocks are swapped and given as input to the next round. After the final round the output if transformed according to the following steps:

(1) Multiply X_1 and the first subkey
(2) Add X_2 and the second subkey
(3) Add X_3 and the third subkey
(4) Multiply X_4 and the fourth subkey

The four sub-blocks are appended to produce the ciphertext.

2.3   GOST

GOST is a 64-bit block encryption algorithm with a 256 bit key. Encryption is performed in 32 rounds where each round uses a different subkey. The plaintext is divided into left (L) and right (R) halves. Let K_i denote the 32 bit subkey for round i. At round i the following operations are performed:

$$L_i = R_{i-1} \tag{1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2}$$

Here, the right half and subkey K_i are added modulo 2^{32}. The result is then divided into eight 4-bit sub-blocks and each sub-block is given as input to a different S-box (8 S-boxes in total). The output of the S-boxes are combined to get a 32-bit block which then is subjected to a 11-bit circular shirt. The result is then XORed with the left half. The left and right halves are then swapped. These operations are repeated for 32 rounds. Decryption is same as encryption.

2.4 RC4

RC4 is a variable key-size stream cipher. The key stream is independent of the plaintext. 256 S-boxes each of size 8×8 is employed. The entries of the S-boxes are permutation of the number 0 to 255 and the permutation are functions of the variable-length key. The key is generated as follows [Schneier 2002]:

$$i = 0; j = 0$$
$$i = (i + 1) \bmod 256$$
$$j = (j + S_i) \bmod 256$$
$$\text{swap } S_i \text{ and } S_j$$
$$t = (S_i + S_j) \bmod 256$$
$$K = S_t \tag{3}$$
$$\tag{4}$$

The plaintext is XORed with K to produce the ciphertext. By XORing K with the ciphertext the plaintext is recovered.

In the next section we describe the hardware and software experimental test-bed set-up that we use to measure the battery power consumption of these encryption algorithms running on a laptop computer along with the mathematical models for the collected data.

3. EXPERIMENTAL SET-UP FOR POWER PROFILING AND MODELLING

3.1 Experimental Set-up and Power Measurement

Clearly, encryption algorithms are computationally complex. They require several rounds of operations, require several CPU cycles, and require a lot of memory. Several factors such as the available memory, hardware architecture, software implementation, etc. have an effect on the final power consumption. The experimental hardware test-bed set up that was used to gather power consumption data for the various encryption algorithms is shown in Figure 2. We used a laptop as the underlying computational device for the experiments. This is because laptop based wireless networking (e.g., IEEE802.11 wireless LAN) is extremely popular currently and is expected to grow rapidly over the next few years.

The set-up consists of a Sony Vaio laptop with a 700 Mhz P-III processor and 128 MB RAM running Red Hat Linux 2.4.8 chosen for its open source nature. The power consumed by the CPU in running the encryption algorithms is measured as a function of input power supply to the Laptop. A separate DC power supply is given to the laptop to permit measurements. The battery of the laptop is removed for accuracy in measurements. The current measurements are gathered using Labview software [lab] from the GPIB interface of the power supply. In order to eliminate effects of the other jobs that could be running in the background the current consumption is first measured when no other tasks are running (idle amps). The difference in currents when an encryption algorithm is running and idle amps is taken as the actual current consumption during encryption. In the experiments, since voltage variation is seen to be extremely small (measured at less than 0.25%) we use a constant value given by the manufacturer. Power consumption value is
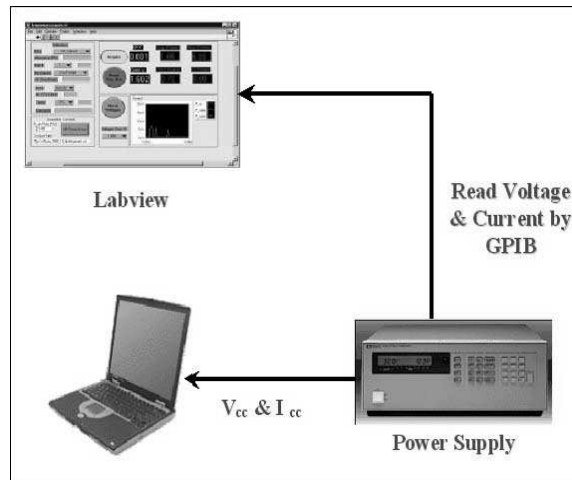
Fig. 2. Experimental set-up for profiling battery power consumption of encryption algorithms.

then computed as the product of the voltage and the current consumption. Several experiments with different data sets were conducted and the average of these results is calculated as the final (average) power consumption value.

To monitor the power consumption of a a software encryption module we use OProfile [opr]. OProfile is a system-wide profiler for Linux systems capable of profiling all running code at low overhead. OProfile leverages the hardware performance counters of the CPU to enable profiling of a wide variety of interesting statistics which can also be used for basic time-spent profiling. All of the code is profiled: hardware and software interrupt handlers, kernel modules, the kernel, shared libraries, and applications. So we have adapted OProfile to monitor the different components of an encryption algorithm in order to measure the power values for the different functions involved. Each encryption algorithm was divided into two portions: *setup functions* that initialize the key elements that would be used in encryption/decryption and *core functions* which repeatedly perform operations on the data block.

3.2 Modelling Power Consumption

In our experiments we used ten random plaintext data sets. These were encrypted and decrypted using the different algorithms and the battery power consumption was profiled. Some experimental data were used for model fitting and the remaining were used for testing the model. From the experiments we observe that in DES the function which involves both data expander and S-box substitution takes almost 75% of total execution time. Similarly for IDEA and GOST the core functions that are carried out every round consume more time. Figure 3 shows a comparison of consumed power for encrypting using DES, IDEA and GOST. From Figure 3 we conclude that the differences in the power consumption values for a fixed key length

and number of rounds is not significant for these three encryption algorithms.
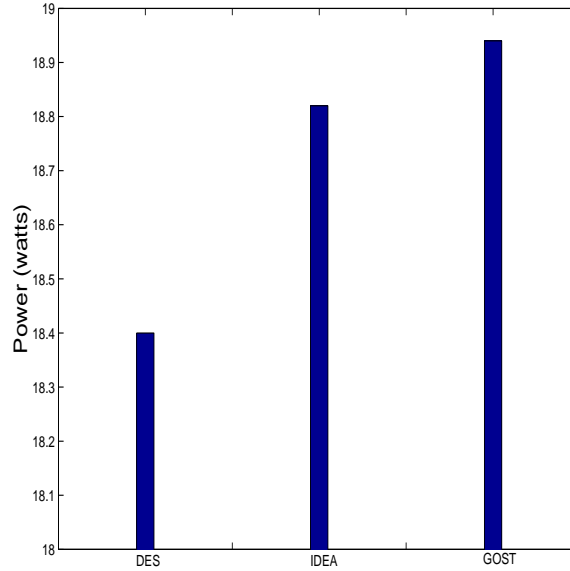

Fig. 3. Power consumption comparison of different block ciphers.

Figure 4 shows the variation of consumed power for different rounds of DES, IDEA and GOST. We observe from these figures that the power consumption varies linearly with the number of rounds. In these figures, half of the data points were obtained experimentally to fit a mathematical model. The remaining data points were obtained using the model to validate the model. Let P denote the consumed power (Watts) and r the number of encryption rounds, respectively. Then using statistical regression, for DES we find that:

$$P(r) = 0.0486r + 17.7335 \tag{5}$$

The standard of error of this model is .0139 meaning the curve fit is a good approximation of the actual behavior. Similarly, for IDEA we find that,

$$P(r) = 0.0975r + 18.015 \tag{6}$$

with a standard of error equal to 0.03427. For GOST, the power consumption model obtained is:

$$P(r) = 0.03321r + 17.90204 \tag{7}$$

with standard of error equal to 0.0450. From these models we see that among the three block ciphers GOST has the smallest slope implying the rate of change of power w.r.t. the number of rounds is the smallest for GOST and maximum for IDEA.
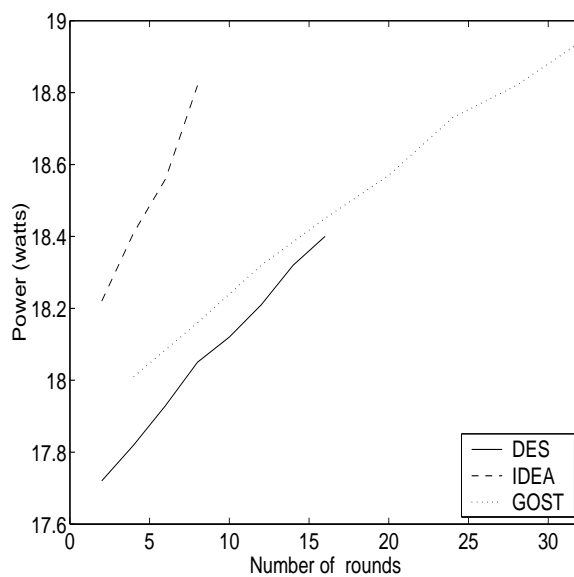
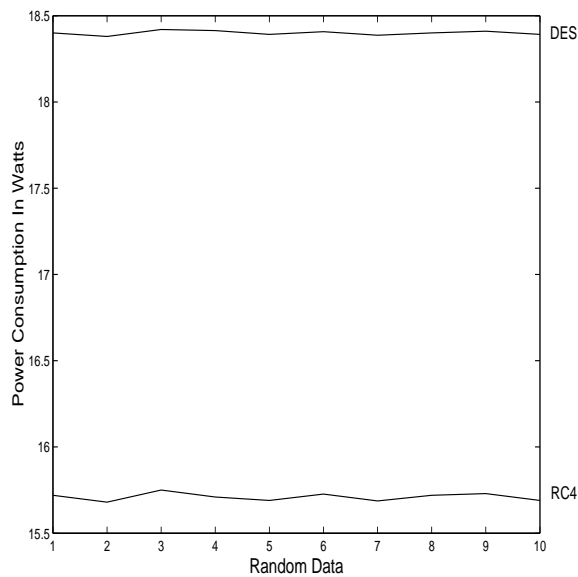Fig. 4.   Power consumption for different rounds of DES, IDEA and GOST.

Fig. 5.   Power consumption comparison of DES and RC4.

Fig. 5 compares the power consumption of 16 round DES and 128-bit key RC4. It is not surprising to see that the difference between the power consumption values is significant. This is because the computations performed in RC4 are simple binary operations. A statistical regression analysis using the power values obtained for RC4 for varying values of the key length (k) produces the following non-linear power consumption model:

$$P(k) = 15.85e^{-3.311/k} \tag{8}$$

Here, we used k=32, 64, 128, 192 and 256 to obtain experimental data for model fitting and key lengths of 96 and 192 to test it. A standard error of 0.0797 was obtained showing that the non-linear model (8) is a good fit.

4. SECURITY OPTIMIZATION, CRYPTANALYSIS AND POWER CONSTRAINT

Symmetric block ciphers are popular in several applications. This popularity requires a high level of trust in their security. Unfortunately there are neither known constructions of block ciphers, which offer unconditional security nor practical constructions, which offer provable computational security. Therefore measuring the security of a cipher is itself an open issue. However, one way to measure the effectiveness of a an encryption algorithm is its resilience to cryptanalysis attacks. There are three popular types of cryptanalysis attacks on encryption algorithms, they are:

—brute-force attack,

—differential cryptanalysis [Biham and Shamir 1991]

—linear cryptanalysis [Matsui 1993; 1994].

In a brute-force attack all possible encryption keys are successively tested to find the correct one. In differential cryptanalysis the differences in the ciphertexts for two chosen pairs of plaintexts are compared. This comparison is used to deduce the key.

Linear cryptanalysis is an attempt to find linear dependency of high probability between the plaintext, the ciphertext and the key, by which the key may be retrieved. This is a known plaintext attack, i.e., the cryptanalyst is assumed to know some plaintexts and the corresponding ciphertexts. The basic idea is to approximate the operation of a portion of the cipher with an expression that is linear w.r.t. mod-2 bit-wise operation.

Note that the number of rounds and key length impact the total power consumption and the security against successful cryptanalysis attacks. Therefore, it is possible to find a mathematical relationship between power consumption and the offered security. One way to measure security is to compare the vulnerability of a cipher to a cryptanalysis (linear, differential etc.) attack against an exhaustive key search. We consider linear cryptanalysis attack of DES [Matsui 1993] for the sake of illustration. Since all the operations in DES except the S-boxes are linear it suffices to derive linear relations of the S-boxes. These relations are derived for each S-box by choosing a subset of input bits and output bits, calculating parity of these bits for each of possible inputs of S-box and counting the number of inputs whose subset parity is zero. As the number of zeros is closer to number of ones we will say that

subset is more nonlinear. We have to find a statistical linear expression consisting of parity of subsets of the plaintext, ciphertext and the key which is derived from similar expressions of various rounds. Thus, the parity of some set of data bits in each round is known as a function of the parity of the previous set of bits in the previous round and parity of several key bits. The round linearization is based on the linearization of S-boxes.

The DES algorithm is vulnerable to linear cryptanalysis attacks. By such an attack, the algorithm in its sixteen rounds can be broken using 2^{47} known plaintexts [Matsui 1993]. This vulnerability raises a notable risk when encrypting bulk data that may be predictable with keys that are constant. Let $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{K}$ stand for the plaintext, ciphertext and the key vector, respectively. Then, following [Matsui 1993], the purpose of linear cryptanalysis is to find the following effective linear expression for an encryption algorithm:

$$\mathcal{P}[i_1, i_2, ..., i_a] \bigoplus \mathcal{C}[j_1, j_2, ..., j_b] = \mathcal{K}[k_1, k_2, ...k_c] \tag{9}$$

where $i_1, i_2, ..., i_a$, $j_1, j_2, ..., j_b$ and $k_1, k_2, ...k_c$ denote fixed bit locations. This equation holds with probability not equal to $1/2$ for a random plaintext $\mathcal{P}$ and the corresponding ciphertext $\mathcal{C}$. Then, the value of $|p - 1/2|$ represents the effectiveness of linear expression in Eq. (9). Let N be the number of given random plaintexts and p be the probability that (9) holds. If $|p - 1/2|$ is sufficiently small then the linear cryptanalysis success rate increases with N or $|p - 1/2|$. This result is formalized in the following theorem given in [Matsui 1993].

THEOREM 4.1. *If $|p-1/2|$ is sufficiently small then the success rate of the linear cryptanalysis algorithm is $\int_{-2\sqrt{N}|p-1/2|}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$.*

From this theorem the success rate of the cryptanalysis algorithm w.r.t. N can be computed as follows:

N	$1/4|p - 1/2|^{-2}$	$1/2|p - 1/2|^{-2}$	$|p - 1/2|^{-2}$	$2|p - 1/2|^{-2}$
Success Rate	84.1%	92.1%	97.7%	99.8%

Using this analysis it can been seen that for an 8-round DES the key is breakable with 2^{21} known plaintexts, 12-round DES in 2^{33} known plaintexts and a 16-round DES with 2^{47} plaintexts [Matsui 1993].

Another recent cipher is the AES [AES]. It is known that currently there is no known attack on full AES. But there have been attempts (e.g., [Biryukov 2004; Keliher 2004]) to attack a reduced round AES. For example, in [Biryukov 2004] it is shown that the complexity of a Boomerang attack on 5-round AES is 2^{38} chosen plaintexts. If further research on attacking full round AES is successful then the data complexity of those attacks may be incorporated into our security-power trade-off framework using an approach similar to that of DES. A modified vulnerability measure may also be introduced.

The vulnerability decreases as the number of DES rounds increases. Therefore we observe that the success probability of a known plaintext linear cryptanalysis attack can be computed as a function of the number of rounds. Based on this observation we define a measure called the *vulnerability* of a cipher as follows.
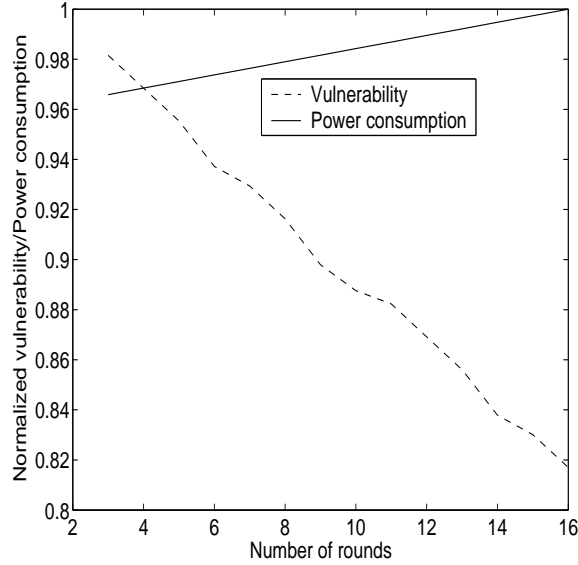
Fig. 6.   Vulnerability-power consumption trade-off in DES for 97.7% cryptanalysis success rate.

Definition 4.2. *Vulnerability* is defined as the ratio of maximum number of plaintexts (for a given block length) required for a brute force search to the number of plaintexts required using a cryptanalysis algorithm to successfully estimate the encryption key bit(s).

Therefore this metric measures the amount of reduction in cryptanalysis complexity (required number of plaintexts) when compared with a brute force search. For example, if we take an 8 round DES it requires a total of 2^{21} plaintexts to estimate the key using linear cryptanalysis. If we take a 64 bit block then there are 2^{64} possible plaintexts. Therefore, in this case vulnerability is given by $\frac{2^{64}}{2^{21}} = 2^{43}$ (note that in reality the numerator for DES must be 2^{56}). Figure 6 shows the (normalized) vulnerability for different number of DES rounds in the $\log_2(.)$ scale. It is clear from this figure that as the number of rounds increases the vulnerability decreases. On the other hand the normalized power consumption (normalized by the maximum power consumption) increases as given in (5).

4.1   Vulnerability Minimization with Power Constraint

Suppose we have M data packets (or class of packets) and that all the packets are not equally important in terms of security (or vulnerability) requirement. This occurs in several applications. For instance, in video applications, the motion vector packets need to be more secure than the packets containing texture information. The question then is: *How do we minimize the total vulnerability subject to a total power constraint, say, P_t?* A simple strategy is to allocate power P_t/M to each

packet. But, this may not be optimal. Therefore, we formulate the problem as a constrained optimization as follows.

4.1.1 *Optimization Formulation 1.* In this subsection we use DES for illustration. The objective is to optimally allocate the battery power resource to M packets with different vulnerability requirement such that the total power budget is not exceeded. Mathematically, this problem is given by,

$$\min_{\{P_1, P_2, \ldots P_M\}} \sum_{k=1}^{M} w_k V_k \quad \text{s.t.} \quad \sum_{k=1}^{M} P_k \leq P_t \tag{10}$$

where V_k stands for normalized vulnerability of packet k, $k = 1, 2, \ldots M$, $0 \leq w_k \leq 1$ is a weighting parameter, and $P_k(r_k)$ is the (normalized) power allocated to encrypt the kth packet. Note that a higher value of w_k implies a higher security requirement for that packet. From the data plotted in Fig. 6 we obtain a regression model for the normalized vulnerability as:

$$V_k(r_k) = -0.0501 r_k + 1.0665 \tag{11}$$

which has a standard error of 0.04. Here $V_k(r_k)$ means that the kth packet is encrypted using r_k rounds. Then using (5) and (11) it is easy to see that we can re-formulate the optimization problem (10) as follows:

$$\min_{\{P_1, P_2, \ldots P_M\}} \sum_{k=1}^{M} w_k [-19.2692 P_k + 19.5264] \quad \text{s.t.} \quad \sum_{k=1}^{M} P_k \leq P_t; \ 0.961 \leq P_k \leq 1 \tag{12}$$

This is equivalent to:

$$\min_{\{P_1, P_2, \ldots P_M\}} -19.2692 \sum_{k=1}^{M} w_k P_k \quad \text{s.t.} \quad \sum_{k=1}^{M} P_k \leq P_t; \ 0.961 \leq P_k \leq 1. \tag{13}$$

The optimal solution to problem (13) is obtained by replacing the inequality constraint $\sum_{k=1}^{M} P_k \leq P_t$ with the equality constraint. It is clear that the solution to (13) can be obtained by solving the following maximization problem:

$$\max_{\{P_1, P_2, \ldots P_M\}} \sum_{k=1}^{M} w_k P_k \quad \text{s.t.} \quad \sum_{k=1}^{M} P_k \leq P_t; \ 0.961 \leq P_k \leq 1. \tag{14}$$

We implicitly assume that $P_t \geq 0.96 M$ so that each packet is at least minimally encrypted. The new problem is equivalent to the *knapsack problem*, a classical optimization problem.

Definition 4.3. [Dantzig 1957; Lawler 1976] There is a knapsack of capacity $W > 0$ and n items. Each item has a value $\pi_i > 0$ and weight $\gamma_i > 0$. Find the selection of items that fit within the knapsack while maximizing the value of items selected. That is, if $x_i = 1$ if item i is selected, then maximize $\sum_{i=1}^{n} x_i \pi_i$ subject to $\sum_{i=1}^{n} x_i \gamma_i \leq W$.

There are two versions of the *knapsack problem*. In the $0-1$ version of the *knapsack problem*, the variable x_i can either be 0 or 1. That is, either the whole item is selected or it is not. In the fractional version of the *knapsack problem*, x_i

is allowed to take fractional values between 0 and 1 inclusive. The $0 - 1$ version is NP-hard, but the fractional version is solvable in polynomial time. According to a folklore algorithm [Dantzig 1957], selecting the items in non-increasing order of maximum value per unit weight (π_i/γ_i), maximizes the value of items selected.

Observe that for every packet k, we should allocate at least 0.961 units of normalized power. In general, let us denote the lower and upper bounds on the power allocation to encrypt each packet as v_k and z_k respectively. Note that in our case $v_k = 0.961, \forall k$ and $z_k = 1, \forall k$. Therefore, we can modify the above formulation as follows:

$$\max_{\{P_1, P_2, \ldots P_M\}} \sum_{k=1}^{M} w_k P_k \quad \text{s.t.} \quad \sum_{k=1}^{M} P_k \leq P_t'; 0 \leq P_k \leq z_k - v_k \qquad (15)$$

where $P_t' = P_t - \sum_{k=1}^{M} v_k$ is the residual number of units of power after allocating v_k units of power (to encrypt) for each packet k. This formulation is equivalent to the fractional version of the *knapsack problem*. Therefore, the following greedy algorithm, referred as **GreedyAlloc_Power**, solves the problem optimally.

GreedyAlloc_Power:

Initialization: Allocate v_k units of power to encrypt packet k, $k = 1 \ldots M$.
Step 1: Sort the M packets in non-increasing order of weight
$\quad\quad$ w_k, $k = 1 \ldots M$.
Step 2: Allocate the additional maximum possible power ($\leq z_k - v_k$)
$\quad\quad$ to each packet k in the sorted order, i.e., $w_k > w_{k+1}$.
$\quad\quad$ That is, $z_k - v_k$ units of power are allocated to
$\quad\quad$ packets $k = 1 \ldots j^* - 1$ for some j^*, 0 or fewer than $z_{j^*} - v_{j^*}$
$\quad\quad$ power is allocated to packet j^* and 0 power is allocated to packets
$\quad\quad$ $k = j^* + 1, \ldots M$ where the sum total of the additional power
$\quad\quad$ allocated is equal to P_t'.

THEOREM 4.4. *Algorithm **GreedyAlloc_Power** is optimal.*

PROOF. We will prove the optimality of **GreedyAlloc_Power** by contradiction. Recall that, **GreedyAlloc_Power**, only deals with the additional power allocation over and above v_k that is allocated to each k.

Let us assume that **GreedyAlloc_Power** allocates P_k units of additional power to packet k as described in the algorithm. Assume there exists an optimal power allocation solution different from the solution of **GreedyAlloc_Power** that allocates O_k units of additional power to packet k. Let i be the first packet where the optimal solution differs from the **GreedyAlloc_Power** solution. Hence, $O_i < P_i$ by virtue of the **GreedyAlloc_Power** algorithm. Therefore for some $j > i$, we should have $O_j > P_j$. Let $\delta = \min\{O_j - P_j, P_i - O_i\}$. In the optimal solution, if we reallocate δ units from j to i, then, the cost of the optimal solution will increase by $\delta \cdot (w_i - w_j)$ since $w_i > w_j$. Therefore, this reallocation gives a solution better than the optimal solution, which is a contradiction. Hence **GreedyAlloc_Power** is optimal. $\square$

4.1.2 *Optimization Formulation 2.* Suppose a relationship between the required number of plaintext-ciphertext pairs and cryptanalysis success rate is not available. Then we can formulate the optimization problem differently as follows. Let r_k denote the number of rounds of encryption allocated for packet k, $k = 1, 2, \ldots, M$. Let $w_k > 0$ be a weighting parameter; a higher value of w_k implies a higher security requirement. Let P_t be the total power constraint. It is easy to see that we can convert P_t into an equivalent upper bound constraint on the total number of rounds R_t of encryption using relations such as (5), (6), (7) etc. The security of the ciphers increase with the number of rounds of encryption (which in turn increases the power consumption). However, due to the power constraints, there is only a limited total number of rounds for which the encryption algorithm can be run on all the M packets. Therefore, the available number of rounds R_t is a limited resource that needs to be judiciously allocated across all the M packets so as to maximize the weighted total security of all the packets. We formulate this problem as an integer linear program with integer variables r_k where r_k is the number of rounds of encryption run on packet k:

$$\max_{\{r_1, r_2, \ldots r_M\}} \sum_{k=1}^{M} w_k r_k \ \text{ s.t. } \ \sum_{k=1}^{M} r_k \leq R_t; l_k \leq r_k \leq u_k \tag{16}$$

where l_k and u_k are the lower and upper bounds on the number of rounds r_k for packet k that are user fixable within the constraints of the encryption algorithm. A higher value of l_k means a higher security for that packet.

Observe that for every packet k, we should allocate at least l_k rounds. Therefore, we can modify the above formulation as follows:

$$\max_{\{r_1, r_2, \ldots r_M\}} \sum_{k=1}^{M} w_k r_k \ \text{ s.t. } \ \sum_{k=1}^{M} r_k \leq R_t'; 0 \leq r_k \leq u_k - l_k \tag{17}$$

where $R_t' = R_t - \sum_{k=1}^{M} l_k$ is the residual number of rounds after allocating l_k rounds for each packet k. We again implicitly assume that $R_t \geq M \max\{l_k\}$ so that each packet is at least minimally encrypted. Note that this problem is very similar to (15). Therefore most of the previous analyses will hold here as well. The problem (17) is again equivalent to the fractional version of the *knapsack problem*. Therefore, the following greedy algorithm, referred as **GreedyAlloc_Rounds**, solves the problem optimally.

GreedyAlloc_Rounds:

Initialization: Allocate l_k encryption rounds to packet k, $k = 1 \ldots M$.
Step 1: Sort the M packets in non-increasing order of weight
$\quad\quad\quad w_k$, $k = 1 \ldots M$.
Step 2: Allocate the additional maximum possible number of rounds $(\leq u_k - l_k)$
$\quad\quad\quad$ to each packet k in the sorted order,
$\quad\quad\quad$ i.e., $w_k > w_{k+1}$. That is, an additional $u_k - l_k$ rounds are allocated for
$\quad\quad\quad$ packets $k = 1, \ldots, j^* - 1$ for some j^*, 0 or fewer than $u_j^* - l_j^*$
$\quad\quad\quad$ rounds are allocated for packet j^* and 0 rounds are allocated for packets
$\quad\quad\quad k = j^* + 1, \ldots M$ where the sum total of the additional number of rounds
$\quad\quad\quad$ allocated is equal to $R_t^{'}$.

THEOREM 4.5. *Algorithm **GreedyAlloc_Rounds** is optimal.*

PROOF. The proof is identical to the optimality proof of **GreedyAlloc_Power**.   □

5. NUMERICAL RESULTS

In this section we describe the numerical results obtained from the theoretical analysis presented in the previous section. We present results only for DES to avoid repetition. However, we note that the same principles can be used for other block ciphers and also to select the optimal key size for RC4.
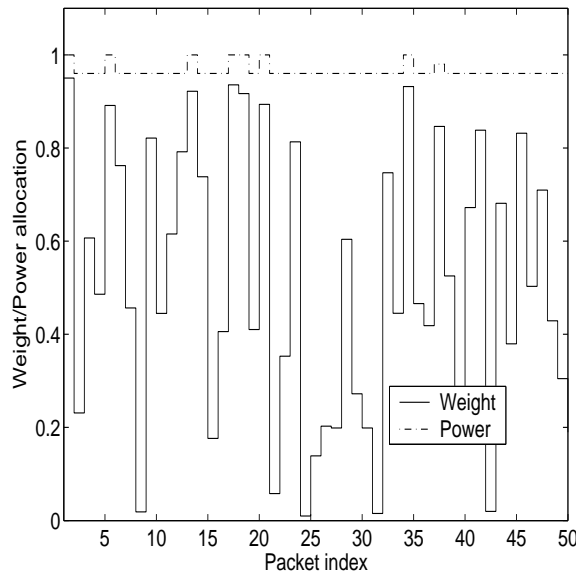

Fig. 7.   Optimal weight dependent normalized power allocation to encrypt packets.

Figure 7 shows the results obtained using the **GreedyAlloc_Power** algorithm when $P_t = 48.3$. The weights for the packets were generated using a uniform random variable between 0 and 1. We see from the figure that packets with higher weights are allocated higher power. The corresponding number of DES encryption rounds can be computed using (5). All the packets are allocated at least 0.96 (lower bound) units of power. The power allocation values varied over 1, 0.98, and 0.96 for the different packets. The weighted average of vulnerability for the optimal allocation is 0.8203 when compared with 0.9124 for equal power allocation. This translates to a gain of 2^6 plaintexts due to optimal power allocation—the cryptanalyst will need a factor of 2^6 additional plaintexts to attain the same performance when equal power allocation is employed. Also, for the most important packet(s) the optimal allocation assigns power equivalent to a 16 round encryption while equal allocation assigns 1 round of encryption for every packet, for the given constraint. This clearly shows that a naive, equal power allocation is not efficient, especially when the power constraint is stringent.
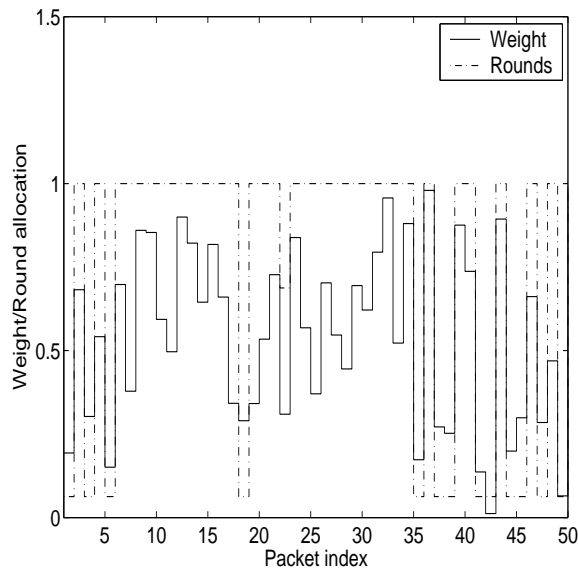

Fig. 8.   Optimal weight dependent normalized round allocation to encrypt packets.

Figure 8 shows the results obtained using the **GreedyAlloc_Round** algorithm when $R_t = 600$. The weights for the packets were again generated using a uniform random variable. It is clear from the figure that packets with higher weights are allocated higher number of encryption rounds. All the packets were allocated at least 1 round and a maximum of 16 rounds. The results show that round allocation values varied over 16, 11, and 1 for the different packets. The weighted average of vulnerability for the optimal allocation is 0.3399 when compared with 0.4653 for

equal round allocation—the cryptanalyst needs a factor of 2^8 additional plaintexts to achieve the same performance as the equal round allocation. We also notice that the optimal allocation assigns 16 rounds to the most important packet(s) while the equal allocation assigns only 12 rounds.

Finally, we chose a scenario when the 50 packets were equally divided into two distinct (high and low) priority classes. The value of l_k was set to 8 for the high priority packets and 1 for the low priority ones. $R_t=300$ was chosen with the weights equal to 0.9 and 0.4 for the high and low priority packets, respectively. We found that the proposed optimal round allocation assigned 16 rounds to many of the high priority packets and the number of assigned round varied over the set $\{16, 11, 8, 1\}$. The equal allocation assigned only 6 rounds for all the packets. Clearly, the high priority packets are more vulnerable to cryptanalysis attack by the equal allocation algorithm.

6. CONCLUSIONS

From the profiled data obtained from running encryption algorithms on a laptop computer it is seen that power consumption changes linearly with the number of rounds of DES, IDEA and GOST encryption algorithms. GOST has the smallest rate of increase of power consumption. The power consumption of RC4 varies non-linearly w.r.t. the key length.

The vulnerability minimization problems subject to upper bounds on the total power and total encryption round indicate that the optimal resource (power/number of rounds) solution is obtained by replacing the inequality upper bound constraint by its corresponding equality constraint. For the examples considered in this paper it is observed that the cryptanalyst will need an additional factor of 2^8 known plaintexts to achieve the same performance of equal resource allocation. This shows that significant gains in terms of security subject to resource constraints can be achieved by using the proposed solutions. If the lower and upper bounds on the resource constraints are different for packets with different priorities then the gains in security by using the proposed method is generally more pronounced.

Acknowledgements

REFERENCES

Labview. *http://www.ni.com*.

Oprofile: A system profiler for linux. *http://oprofile.sourceforge.net*.

BAPATLA, S. AND CHANDRAMOULI, R. 2004. Battery power optimized encryption. *Proc. IEEE Intern. Conf. on Communications 7*, 3802–3806.

BIHAM, E. AND SHAMIR, A. 1991. Differential cryptanalysis of des-like cryptosystems. *Journal of Cryptology 4*, 1, 3–72.

BIRYUKOV, A. 2004. Boomerang attack on 5- and 6-round aes. *Fourth Conference on the Advanced Encryption Standard (AES)*, 11–15.

BUCHMANN, I. Batteries in a portable world. *http://www.buchmann.ca/*.

DANTZIG, G. B. 1957. Discrete-variable extremum problems. *Operations Research 5*, 266–277.

DES. 1977. National bureau of standards. data encryption standard. *U. S. Department of Commerce*.

GOODMAN, J. AND CHANDRAKASAN, A. 1998. Low power scalable encryption algorithms. *Wireless Networks The Journal of Mobile Communication, Computation and Information 4,* 1, 55–70.

GOODMAN, J., CHANDRAKASAN, A., AND DANCY, A. 1999. Design and implementation of a scalable encryption processor with embedded variable dc/dc converter. *Proc. of the 36th ACM/IEEE Conference on Design Automation*, 855–860.

GOST. 1989. Gosudarstvennyi standard 28146-89. *Cryptographic Protection for Data Processing Systems*.

HODJAT, A. AND VERBAUWHEDE, I. 2002. The energy cost of secrets in ad-hoc networks. *IEEE Circuits and Systems Workshop on Wireless Communications and Networking*.

IEEE802.11B. *http://grouper.ieee.org/groups/802/11/*.

KARRI, R. AND MISHRA, P. 2003. Minimizing the secure wireless session energy. *Journal of Mobile Network and Applications (MONET) 8,* 2 (April), 177–185.

KELIHER, L. 2004. Refined analysis of bounds related to linear and differential cryptanalysis for the aes. *Fourth Conference on the Advanced Encryption Standard (AES) "AES - State of the Crypto Analysis"*, 42–57.

LAI, X. 1992. On the design and security of block ciphers. *ETH Series in Information Processing 1*.

LAWLER, E. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston.

MATSUI, M. 1993. Linear cryptanalysis method for des cipher. *Advances in Cryptology - Eurocrypt*, 386–397.

MATSUI, M. 1994. The first experimental cryptanalysis of data encryption standard. *Advances in Cryptology Crypto'94*, 1–11.

MICHELL, S. AND SRINIVASAN, K. 2004. State based key hop protocol: a lightweight security protocol for wireless networks. *Proc. of 1st ACM International Workshop on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Network*, 112–118.

MPEG4. *http://www.m4if.org/*.

NECHVATAL, J., BARKER, E., DODSON, D., DWORKIN, M., FOTI, J., AND ROBACK, E. 1999. Status report on the first round of the development of the advanced encryption standard. *Journal of Research of the National Institute of Standards and Technology 104,* 5, 435–459.

PERRIG, A., SZEWCZYK, R., TYGAR, J. D., WEN, V., AND CULLER, D. 2002. Spins: Security protocols for sensor networks. *ACM Journal of Wireless Networks 8,* 5 (Sept.), 521–534.

POTLAPALLY, N., RAVI, S., RAGHUNATHAN, A., AND JHA, N. 2003. Analyzing the energy consumption of security protocols. *International Symposium on Low Power Electronics and Design*, 30–35.

PRASITHSANGAREE, P. AND KRISHNAMURTHY, P. 2003. Analysis of energy consumption of rc4 and aes algorithms in wireless lans. *Proc. IEEE Global Telecommunications Conference 22,* 1 (Dec.), 1445–1449.

RAVI, S., RAGHUNATHAN, A., AND POTLAPALLY, N. 2002. Securing wireless data: System architecture challenges. *Proc. Int. Symp. System Synthesis*, 195–200.

RIVEST, R. 1992. The rc4 encryption algorithm. *RSA Data Security, Inc.*.

SCHNEIER, B. 2002. *Applied cryptography : Protocols, algorithms, and source code in C*. Vol. 2. John Wiley and Sons.

SKLAVOS, N. AND KOUFOPAVLOU, O. 2001. Asynchronous low power vlsi implementation of the international data encryption algorithm. *Proc. of IEEE International Conference on Electronics, Circuits and Systems*.

SLIJEPCEVIC, S., POTKONJAK, M., TSIATSIS, V., ZIMBECK, S., AND SRIVASTAVA, M. 2002. On communication security in wireless ad-hoc sensor networks. *11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.*, 139–144.

TOSUN, A. AND FENG, W. 2001. Lightweight security mechanisms for wireless video transmission. *Proc. IEEE International Conference on Information Technology: Coding and Computing*, 157–161.