Systematic Security and Timeliness Trade-Offs in Real-Time Embedded Systems

Kyoung-Don Kang
Department of Computer Science
State University of New York at Binghamton
kang@cs.binghamton.edu

Sang H. Son
Department of Computer Science
University of Virginia
son@cs.virginia.edu

Abstract

Real-time embedded systems are increasingly being networked. In distributed real-time embedded applications, e.g., electric grid management and command and control applications, it is required to meet not only real-time but also security requirements. It is essential to meet as many deadlines as possible, e.g., to avoid a power outage or loss of a life. It is also necessary to support the confidentiality and integrity of the data to ensure a potential adversary cannot read transmitted data or corrupt them without being detected. Unfortunately, in general, cryptographic functions are computationally expensive, possibly causing deadline misses in real-time embedded systems with limited resources. As a basis for cost-effective security support in real-time embedded systems, we define the notion of Quality of Protection (QoP). Further, we present an adaptive security policy in which the QoP is degraded by decreasing the cryptographic key length for certain tasks, if necessary, to improve the success ratio under overload conditions. Although adaptive security support in real-time systems has previously been studied, our work is different in that we can quantify the degree of security degradation and specify the period of time for which a shorter key is acceptable. Our adaptive security policy is also integrated with feedback-based utilization control to find the degree of security adaptation, while avoiding potential overloads even given dynamic workloads. In the performance evaluation, we show that our approach can considerably improve the success ratio at the cost of controlled QoP degradation when overloaded.

1 Introduction

Real-time embedded systems, which used to be isolated, are increasingly being inter-networked due to distributed real-time embedded (DRE) applications including, e.g., electric grid management, agile manufacturing, and defense applications. In these applications, meeting deadlines is essential, e.g., to avoid a power outage or loss of a life. In addition to hard real-time sensing and control, DRE systems need to report the real world status, e.g., the electric grid or battle field status, to the control center that manages the overall electric grid or plans battle tactics. Therefore, it is required to support the confidentiality and integrity of the data transmitted across the network to ensure that a potential adversary cannot read or corrupt the data without being detected.

Unfortunately, most cryptographic algorithms are computationally expensive, possibly causing many deadline misses in real-time embedded systems with limited resources. Note that resource over-provisioning may not be a viable solution due to stringent cost, size, weight, and power constraints prevalent in these systems. On the other hand, it is not desirable to ignore security requirements or simply use a weak security scheme all the time. Thus, it is necessary to balance timing and security requirements. Despite the importance of the problem, relatively little work has been done.

To shed light on the problem, we present a new approach called SSTT (Systematic Security and Timeliness Trade-offs) in real-time embedded systems. We first describe the security requirements in real-time embedded applications and define our research goal. Specifically, we aim to maximize the success ratio, i.e., the fraction of the submitted tasks finishing within their deadlines, while meeting the confidentiality and integrity requirements. Second, we define a new *quantitative* metric to measure the Quality of Protection [9]. Generally, the quality of security service can only be measured qualitatively unlike real-time performance. Even the advanced notion of Quality of Protection (QoP) first introduced in [9] is not quantitative but a qualitative metric. In this paper, we provide a quantitative QoP metric in terms of cryptographic key length to measure the QoP in real-time embedded systems. Based on the redefined QoP concept, we present an adaptive security policy in which the QoP is degraded by decreasing the cryptographic key length for certain tasks, if necessary, to improve the success ratio under overload conditions. Although adaptive security support in real-time systems has previously been studied [20, 3, 19, 20, 6], our work is different in that we can quantify the degree of security degradation and specify the period of time for which a shorter key is acceptable. (The system should switch to the original, long key before the period expires.)

Our adaptive security policy is seamlessly integrated with feedback control to avoid overloads by controlling the CPU utilization in the feedback loop even in the presence of dynamic workloads. The feedback controller computes the workload adjustment required to meet the target utilization, e.g., 90%. According to the control signal, SSTT can adapt the QoP, if necessary, to improve the success ratio by avoiding overloads. In addition, admission control is applied to incoming tasks, if the system is still overloaded after QoP degradation. By degrading the QoP in a secure manner before admission control, we can further improve the success ratio.

To evaluate the performance, we compare our approach, via an extensive simulation study, to several baseline approaches well accepted for real-time computing. Our approach considerably improves the success ratio even given dynamic workloads, while adapting the QoP in a controlled manner, if necessary, to improve the success ratio when overloaded.

The remainder of this paper is organized as follows. In Section 2, an application scenario is discussed to motivate our work. Further, the scope of the work is described by formulating the problem of security and timeliness management. The relation between the cryptographic key length and strength of defense is discussed in Section 3. Based on the discussion, the feedback control and QoP adaptation are also discussed. In Section 4, the performance of SSTT is compared, via a simulation study, to several baseline approaches applying well accepted open loop and closed loop scheduling principles. Related work is discussed in Section 5. Finally, Section 6 concludes the paper and discusses the future work.

2 Scope of the Work

In this section, a (simplified) application scenario and the security model are discussed to specify the scope of the work. Based on the discussions, the problem of the cost-effective support for security and timing constraints is formulated.

2.1 Application Scenario

In time-critical target tracking [11], for example, UAVs (Unmanned Aerial Vehicles) have hard real-time, e.g., engine thrust control, tasks. In addition, they are required to perform soft real-time reconnaissance tasks for monitoring and transmitting the battle field images to the command and control center (CC). Similarly, DRE systems used in electric grids perform hard real-time sensing and control tasks, while reporting the local grid status to the control center across the network.

In general, the workload characteristics, e.g., execution times and periods, of hard real-time tasks are known *a priori*, while soft real-time workloads may vary in time. When a UAV enters the current area of interest (AOI), for example, it can be required to increase the resolution of the surveillance image and report more frequently. Further, the image processing workload can vary in time. As another example, the CC of an electric grid may request DRE systems in the AOI showing abnormal electricity supply patterns report the status more frequently.

In this paper, we focus on maximizing the success ratio of soft real-time tasks in an individual real-time embedded system dedicated to handling soft real-time tasks, while supporting confidentiality and integrity requirements. By achieving this goal, the overall timeliness and security of DRE applications can be improved. Although we expect our adaptive security policy can also improve the end to end delay, network QoS management is beyond the scope of this paper. A thorough investigation is reserved for future work.

2.2 Security Model

We assume that isolated real-time embedded systems (RTESs), e.g., working in a UAV, are trusted and tamper proof. Thus, a RTES only has to encrypt (and authenticate) data before transmitting them across the network.

We consider a symmetric key system in which a CC and RTES share a secret key, since the encryption/decryption in a public key system takes several orders of magnitude longer than that in a symmetric key system [18]. In this paper, we consider the following cryptographic security support:

- Confidentiality: By encrypting messages, we can prevent a potential adversary, without the secret key shared between the CC and a RTES, from reading the message.
- Integrity: To support the integrity of the message M, a one-way hash value $H(M)$ (where $H(\cdot)$ is the one-way hash function) can be sent in addition to M. The receiver can verify a received message M' by computing the hash value $H(M')$. If the message has not been altered during the transition, the message will be verified successfully, i.e, $H(M') = H(M)$.
- Authenticity: Unfortunately, a one-way hash function itself cannot authenticate the origin of a message. An adversary can fabricate a message and compute the

corresponding hash value, since one-way hash functions are known to the public [18]. A keyed one-way hash function using a secret key between a RTES and CC can handle this problem. Without the shared secret key, an adversary cannot compute the appropriate hash value. Hence, authenticity is a stronger concept than integrity. Using a keyed one-way hash function, one can verify not only the integrity but also the authenticity of messages. Therefore, in the remainder of this paper, we focus on message confidentiality and authenticity. We assume that each RTES, e.g., a RTES in a UAV, shares a pair of secret keys with a CC to encrypt and authenticate messages to support the message confidentiality and authenticity. We use different keys for encryption and authentication, since the rule of thumb is to use separate keys for different purposes [18]. Thus, only the CC can decrypt and check the integrity and authenticity of the message transmitted by the RTES using the shared keys and vice versa.

Finally, it is assumed that main attacks against a scrutinized cryptosystem without any known vulnerability, e.g., DES (Digital Encryption Standard) [18] or AES (Advanced Encryption Standard) [15] are *brute-force attacks* that try to find the secret key via an exhaustive search in the key space as common in trusted cryptosystems [4, 18, 22]. Other attacks such as denial of service attacks are beyond the scope of this paper.

2.3 Problem Formulation

In our model, a (soft) real-time task T_i is associated with a relative deadline D_i. If it is a periodic task, we assume its deadline is equal to the period. An aperiodic soft real-time task is also associated with a relative deadline.

The estimated execution time of a soft real-time task T_i is: $C_i = C_{i,c} + C_{i,e(l_i)}$ where $C_{i,c}$ is the estimated real-time function execution time and $C_{i,e(l_i)}$ is the estimated time for data encryption and authentication when the current key length used by T_i is l_i. Thus, the estimated utilization of a real-time task T_i is: $U_i = C_i/D_i$. As discussed before, the actual execution times of soft real-time tasks may vary, e.g., due to tactical or energy supply reasons.

We aim to maximize the success ratio of soft real-time tasks, while supporting the security requirements discussed before:

$$\text{Maximize } Success\ Ratio = N_t/N_s \tag{1}$$

where N_t and N_s represent the number of the timely tasks that finish within the deadlines and the number of the tasks submitted to the system, respectively.

The success ratio maximization is subject to:

$$l_i \geq l_{i,min} \tag{2}$$

and

$$\sum_{i=1}^{N} U_i \leq B \tag{3}$$

where $l_{i,min}$ is the minimum key length allowed for T_i, N is the number of the tasks currently in the system, and B is the utilization bound of the employed real-time scheduling algorithm such as EDF (Earliest Deadline First) [10].

When Eq 2 is satisfied, we define the QoP as follows:

$$QoP = \frac{\sum_{i=1}^{N} l_i}{\sum_{i=1}^{N} l_{i,max}} \tag{4}$$

where $l_{i,max}$ is the maximum key length that can be used by T_i.

3 Systematic Security and Timeliness Trade-Offs

In this section, the strength of defense estimated by the key length is discussed. Based on the discussion, our approach for systematic security and timeliness trade-offs and its integration with feedback control are discussed.

3.1 Strength of Defense

The strength of a scrutinized symmetric key system, which has no known shortcut to break it, is often estimated by the difficulty of finding the key via brute-force attacks as discussed before. The speed of a brute-force attack, in which an adversary tests all possible keys, is mainly determined by the number of possible key values to be tested and the speed of the key cracking machine(s). For example, when the key is l bits long and the adversary can test m keys per second, it will take him, in average, $2^{l-1}/m$ seconds to find the key.

Wiener [22] designed a specialized parallel hardware to estimate the average time needed for a brute-force attack to break the DES [18] algorithm. Note that this approach is not limited to the DES, but generally applicable to other encryption algorithms such as the AES [15] that is a new encryption standard adopted by the US government. Table 1 shows the average time estimates for hardware brute-force attacks in 2005. (These estimates extrapolate the previous results [18, 22] according to Moore's law.)

Although using a longer key is safer as shown in Table 1, message encryption (or authentication) using a longer key usually takes more time [4], incurring deadline misses. For example, the execution of the AES algorithm using 128, 192, and 256 bit keys in a low-end microprocessor takes approximately 2ms, 3ms, and 4ms in average [5]. Therefore, we propose to use a longer key under light load, while

Table 1. Estimated Average Times for Parallel Brute-Force Attacks

	Key Length		
Hardware Cost	64 bits	80 bits	128 bits
\$10K	37 days	7000 years	10^{18} years
\$100K	4 days	700 years	10^{17} years
\$1M	9 hours	70 years	10^{16} years
\$10M	1 hour	7 years	10^{15} years
\$100M	5.4 minutes	245 days	10^{14} years

switching to a shorter key when the system suffers transient overloads. For example, a RTES can normally use a 128 (256) bit key for the DES (AES) algorithm, while switching to a 80 (128) bit key when overloaded. In addition to improving the timeliness under transient overload by reducing the key size, SSTT has several desirable security and system features as follows:

- A security officer can specify the time period for which a short key can safely be used. For example, an 80 bit DES key can be used in a UAV under transient overload[1] without compromising security as long as the sum of the overload periods is not longer than the period of the key renewal, which can be decided based on Table 1 and the criticality of the specific application. For example, the security officer of a time critical target tracking application [11] can assume the most powerful brute-force adversary (with enough monetary budget) and schedule an 80 bit key renewal every month in a hanger in addition to regular maintenance.[2]

- By switching between several independent keys, we can require an adversary to spend more time to find the key. Also, the key may not be used anymore when the adversary eventually finds it. In general, perfect security is not possible. Therefore, most existing security schemes aim to force a potential adversary to spend more time and resource as we do in this paper.

- Storing a few more keys with different lengths in a RTES does not significantly increase the memory requirement, which is desirable in resource constrained RTESs. Also, key length adaptation incurs little overhead, while providing desirable security features as discussed above.

[1]If the system is consistently overloaded, more hardware resources must be added to support timing constraints.

[2]Alternatively, a RTES can use the AES algorithm with keys that are at least 128 bits long, while using a 128 bit key under overload. In this case, the keys may not have to be renewed for the lifetime of a RTES, e.g., a RTES in a UAV. A thorough investigation of general key renewal issues in RTESs is reserved for future work.

Note that the key length synchronization between a RTES and the CC can be performed in an efficient manner. Before a RTES changes its key length, it can inform the CC by setting the special flag included in a regular message to be encrypted. If it does not receive the acknowledgment from the CC, it retransmits the message for a certain number of times, similar to common transmission protocols. Thus, key length synchronization does not incur extra communication overheads. Also, the probability of successful synchronization is relatively high. For example, when the message loss probability is as high as 0.5 and the probability of losing a message during the transit is independent of other potential message losses, the message will be delivered with the probability higher than 0.98 when it is resent six times. In our approach, a RTES continues to use the original, long key when the synchronization fails after the predefined number of retransmissions at the cost of the reduced success ratio.

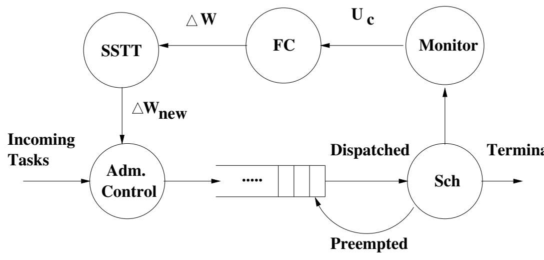

Figure 1. Overall Architecture

3.2 Feedback Control and Security Adaptation

As shown in Figure 1, in our approach, admission control is applied to incoming real-time tasks. Admitted tasks are scheduled by the basic scheduler−the EDF scheduler in this paper. The monitor continuously observes the system behavior in terms of utilization and QoP. The feedback controller (FC) computes the workload adjustment ΔW required to support the target utilization, e.g., 90%, based on the current utilization error E_u, i.e., the difference between the target utilization U_t and current utilization U_c measured at the current sampling period as shown in Figure 2. According to ΔW, SSTT adapts the QoP, if necessary, to reduce the workload under overload.

Interactions between SSTT and feedback control is summarized as follows and discussed in detail in the remainder of this section.

1. At a sampling period, the feedback controller shown in Figure 2 computes ΔW based on the current utilization error $E_u = U_t - U_c$.

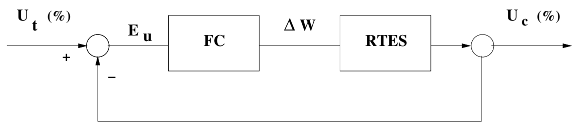

Figure 2. Feedback Control of Utilization

2. If $\Delta W < 0$, SSTT decreases the key length of a task T_i currently in the system when T_i's current key length $l_i > l_{i,min}$. After the QoP degradation, ΔW is adjusted to consider the corresponding workload reduction. Repeat this step until ΔW_{new}, i.e., the new ΔW after the QoP degradation, becomes greater than or equal to zero or every task has become to use its shortest key.

3. If further workload reduction is required after all possible QoP degradation, apply admission control to incoming tasks.

To control the utilization to be below the specified bound U_t, we first model the utilization at the k^{th} sampling instant in the following difference equation:

$$u(k) = \sum_{i=1}^{n} a_i u(k-i) + \sum_{i=1}^{n} b_i \delta w(k-i) \qquad (5)$$

where $u(k-i)$ and $\delta w(k-i)$ are the utilization and workload increase/reduction due to new task arrivals, admission control, and/or QoP adaptation at the $(k-i)^{th}$ sampling instant.

By taking the z-transform [16] of Eq 5, we can algebraically derive the transfer function showing the relation between the input, i.e., the workload adjustment, and output, i.e., the utilization, of the controlled RTES as follows:

$$T(z) = \frac{b_1 z^{n-1} + b_2 z^{n-2} + ... + b_n}{z^n - a_1 z^{n-1} - ... - a_n} \qquad (6)$$

because the z-transform of $u(k-i)$ and $\delta w(k-i)$ are $z^{-i}U(z)$ and $z^{-i}\Delta W(z)$. In this paper, we set $n = 2$ and apply the system identification technique based on the least square method [16] to derive the a_i's and b_i's in Eq 6 ($1 \leq i \leq 2$) based on the simulated workloads described in Section 4. Specifically, we increased the load applied to the system by 50% to 200% by 10% without applying admission control and QoP adaptation to identify the system in the worst tested case.

Based on the system identification results, we applied the Root Locus method [16] to tune the PI (Proportional and Integral) controller used in this paper.[3] The controller is stable since we picked the poles, i.e., the roots of the denominator of the transfer function in Eq 6, inside the unit circle. The sampling period is set to 5 sec to observe the system behavior before adapting the QoP, which is as critical as the real-time performance. The corresponding expected overshoot is 16.2% and settling time is 70 sec, i.e., 12 sampling periods. Generally, the overshoot and settling time have a trade-off relation. We intend to minimize the overshoot that can be more detrimental; a small overshoot can readily be handled in the feedback loop via QoP adaptation and admission control.

[3]We have also considered $n = 3$ in Eq 6 and other controllers such as the PID controller, but we have not observed a large difference in terms of real-time performance.

When $\Delta W < 0$, the QoP is degraded by decreasing the key length for a subset of the tasks currently in the system. When the key length is decreased from l_i to $l'_i (\geq l_{i,min})$ for task T_i, the estimated reduction of the execution time is:

$$\delta C_i = C_{i,e(l_i)} - C_{i,e(l'_i)} \qquad (7)$$

where $C_{i,e(l'_i)}$ is the execution time of the cryptographic function(s) used by T_i after decreasing the key length. The estimated workload reduction due to the QoP degradation is approximately $\delta C_i / D_i$ where D_i is the relative deadline of T_i. After the degradation, we adjust the required workload reduction: $\Delta W_{new} = \Delta W + \delta C_i / D_i$. This QoP degradation procedure is repeated for the other tasks in the system until $\Delta W_{new} \geq 0$ or no more QoP degradation is allowed. Admission control is applied to incoming tasks when $\Delta W_{new} < 0$ and the QoP cannot be degraded anymore. Therefore, newly incoming tasks will not be admitted until ΔW becomes positive as the tasks currently in the system terminate. Overall, our approach is lightweight. The time complexity of feedback control is O(1). When each RTES shares a constant number of keys with the control center, the QoP degradation is O(N) where N is the number of tasks currently in the system.

4 Performance Evaluation

In this section, we describe the baselines, experimental settings, and the simulation results comparing the performance of SSTT to the baseline approaches in the presence of dynamic workloads. A simulation run executes for 10 (simulated) minutes. For each performance data, we take an average of 10 simulation runs executed using different seed numbers. We have also derived the 90% confidence intervals. However, we do not show the confidence intervals, because most of them are smaller than 3%.

4.1 Baselines and Workload Generation

In this paper, we consider several baselines, which are well accepted for real-time scheduling, as follows.

- `EDF`: This approach applies the EDF scheduling policy. All incoming tasks are admitted, while the QoP is not degraded.

- `EDF-AC`: This approach is similar to EDF except it applies admission control to incoming tasks to avoid overloads.

- `FC-AC`: This approach applies EDF scheduling policy and admission control. In addition, it applies the feedback-based utilization control discussed in Section 3. However, the QoP is not degraded in this approach.

Table 2. Simulation Settings

Parameter	**Value**
EET_i	$Uniform(2ms, 10ms)$
AET_i	$(1 + EstErr)EET_i$
$EstErr$	0, 0.2, 0.4, 0.6, 0.8, 1
$Load$	$50\%, 100\%, 150\%, 200\%$
Slack Factor	(10, 20)
#Keys	2
$\lvert$Short Key$\rvert / \lvert$Long Key$\rvert$	0.5
Exec. Time Savings (*ETS*)	10%, 20%, 30%, 40%, 50%

Table 2 summarizes our simulation settings. To generate workloads, we create multiple workload sources that generate tasks whose inter-arrival times are exponentially distributed.[4] Specifically, a task T_i generated by a source S_i is associated with the `estimated execution time` EET_i that is uniformly selected in the range (2ms, 10ms). We consider the relatively long execution time, since RTESs may have to process, e.g., reconnaissance images, while encrypting and authenticating messages where a message can contain a numeric data or a fraction of an image packetized for transmission.

T_i's `actual execution time` is: $AET_i = (1 + EstErr)EET_i$ where the `execution time estimation error` $EstErr$ is varied from 0 to 1 as shown in Table 2. Thus, $AET_i = EET_i$ when $EstErr = 0$, while $AET_i = 2EET_i$ when $EstErr = 1$. As the $EstErr$ increases, it becomes harder to meet timing constraints due to possible errors in admission control, if applied. Note that all the tested baselines and our approach are not aware of $EstErr$. Thus, the system could be overloaded when too many tasks are admitted due to the estimation error.

[4]We have also performed experiments using periodic workloads. In this paper, we only present the experiments using aperiodic workloads in which it is usually more challenging to support timing constraints.

The `relative deadline` of T_i is: $D_i =$ slack factor $\times EET_i$ where the slack factor is uniformly selected in the range (10, 20). The load generated by S_i is equal to AET_i/D_i. The total generated workload is equal to $\sum_{i=1}^{s} AET_i/D_i$ where s is the number of the generated sources. Specifically, we present the performance evaluation results for 50%, 100%, 150%, and 200% loads to observe the system behavior under light load and overload conditions.

Without losing the general applicability of our approach, we assume that a RTES stores two keys where the length of the short key, e.g., 128 bits in the AES algorithm, is a half the length of the long key, e.g., 256 bits. Also, we model the `execution time savings` ETS due to a possible decrease of the length of the key used by T_i. For example, when the ETS is 10%, $AET_{i,new} = 0.9AET_i$ where $AET_{i,new}$ is the actual execution time of T_i after the QoP degradation.

Table 3. Experimental Sets

Set	**Varied**	**Fixed**
1	*Load* $= 50\%, 100\%, 150\%, 200\%$	$EstErr$ = 0 ETS = 10%
2	$EstErr = 0 - 1$	*Load* $= 200\%$ $ETS = 10\%$
3	$ETS = 10\% - 50\%$	*Load* $= 200\%$ $EstErr = 1$

In this paper, among the sets of experiments we have performed, we present the most representative ones shown in Table 3. In the Experiment Set 1, the load applied to the system is increased, while assuming zero $EstErr$ and 10% ETS. Note that this workload favors the baseline approaches, since, for example, EDF-AC's admission control can be precise due to the zero $EstErr$, while SSTT may not be able to significantly improve the success ratio compared to the baselines due to a relatively small ETS. In the Experiment Set 2, we observe the resilience of the tested approaches by increasing the $EstErr$ from 0 to 1, while setting $Load = 200\%$ and $ETS = 10\%$. In the Experiment Set 3, we increase the ETS from 10% to 50%, while setting $Load = 200\%$ and $EstErr = 1$ to observe the potential impacts of ETS on the success ratio and QoP in the presence of high loads and large estimation errors.

4.2 Experiment Set 1: Increasing Load

Figure 3 shows the success ratio when the load increases from 50% to 200% when $EstErr = 0$ and $ETS = 10\%$ as described before. When the load is 50%, every approach achieves the 100% success ratio. As the load increases, the success ratio of EDF drops significantly. It drops to near

31% when the 200% load is applied to the system. This is because EDF simply admits all incoming tasks incurring a lot of backlog and cascading deadline misses.

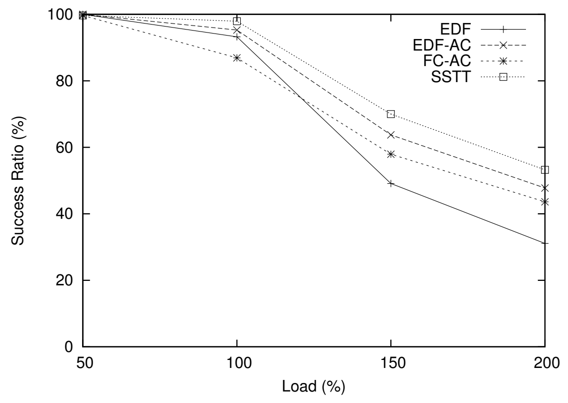

Figure 3. Load vs. Success Ratio

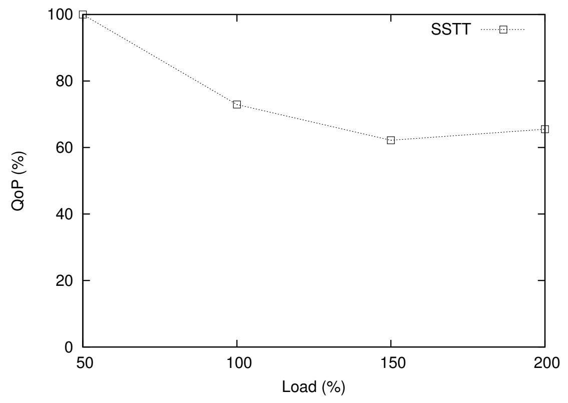

Figure 4. Load vs. QoP

In Figure 3, EDF-AC shows good performance; its success ratio is slightly above 47% when the load is 200%. It improves the success ratio by approximately 16% compared to EDF, because admission control is effective when $EstErr = 0$. The success ratio of FC-AC is slightly lower than EDF-AC. When the load is 200%, it is 43.5%. This is because FC-AC ensures the utilization $\leq$ 90% via feedback-based admission control. Through all the experiments performed, FC-AC and SSTT achieve the target utilization, i.e., 90%, except when the applied load is lower than 90%. In contrast, the CPU is saturated in the open loop approaches, i.e., EDF and EDF-AC, when the load $\geq 100\%$. Hence, we do not discuss the average utilization in the remainder of the paper.

In Figure 3, SSTT achieves the highest success ratio at the cost of the QoP degradation under overload. It achieves the approximately 53% success ratio when the load is 200%, improving the success ratio by more than 20% compared to EDF. As shown in Figure 4, the QoP of SSTT ranges between $60\% - 100\%$. (We do not plot the baselines' QoP, since they do not degrade the QoP regardless of the system status.) However, SSTT only degrades the QoP for a certain period of time that is not long enough for a possible adversary to find the cryptographic key via a brute-force attack as discussed in Section 3. In this experimental set, our approach meets the desired overshoot and settling time described in Section 3.2. The transient performance is discussed in detail in the following subsections dealing with more dynamic workloads.

4.3 Experiment Set 2: Increasing Execution Time Estimation Error

In this section, we apply the 200% load and assume the ETS due to QoP degradation is only 10%, while increasing the EstErr as summarized in Table 3. We also drop EDF, since it has shown the relatively poor performance in the previous section.

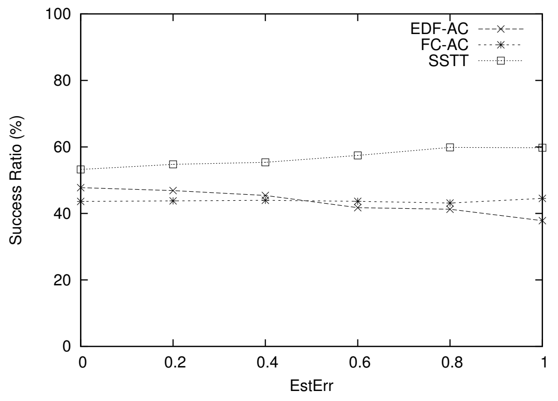

Figure 5. Estimation Error vs. Success Ratio

In Figure 5, as the EstErr increases, EDF-AC's success ratio decreases from 47% to 37%, because EDF-AC erroneously admits more tasks than schedulable due to the increasing EstErr. In contrast, FC-AC's success ratio is stable between $43\% - 45\%$, since the workload is dynamically adjusted via feedback-aided admission control. As shown in the figure, when EstErr = 1, SSTT improves the success ratio by approximately 23% and 16% compared to EDF-AC and FC-AC, respectively. It also improves its own success ratio by approximately 7% as the EstErr increases from 0 to 1. A possible reason is that the QoP degradation may become more aggressive as the EstErr and the corresponding utilization error measured in the feedback loop increases. This observation is supported by Figure 6 in which the QoP of SSTT decreases from approximately 65% to 56% as the EstErr increases.

Figure 7 shows the transient performance of SSTT. The

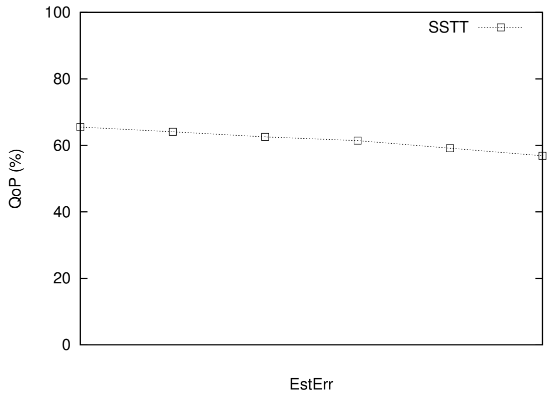

Figure 6. Estimation Error vs. QoP

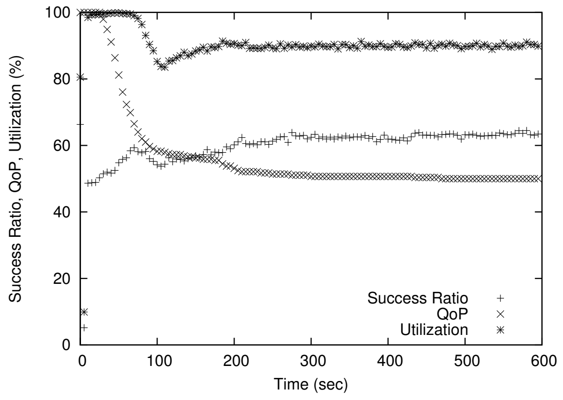

Figure 7. Transient Performance of SSTT (ETS = 10%)

transient success ratio of SSTT increases along the time axis as the initial overload is handled in the feedback loop via QoP adaptation and admission control. The transient transient utilization is lower than or equal to the target utilization, i.e., 90%, except the initial overshoot. The utilization is approximately 98.5% at 10 sec, while it is 90.3% at 90 sec. Therefore, the overshoot is handled in 80 sec, closely meeting the desired settling time, i.e., 70 sec, specified in Section 3.2. In other words, the desired settling time is exceeded by two sampling periods. A further optimization is reserved for future work. Overall, we can observe that potential overloads due to the 200% load and high EstErr, i.e., 1, can be gracefully handled via feedback-based QoP adaptation and admission control, even though the ETS is only 10% in this set of experiments.

4.4 Experiment Set 3: Increasing Execution Time Savings

In this experimental set, we increase the fraction of the execution time saved due to QoP degradation, while setting load = 200% and EstErr = 1. Note that, for the same amount of the key length decrease, the execution time savings may vary for several reasons such as the complexity of the specific cryptographic algorithm used to encrypt and authenticate messages.

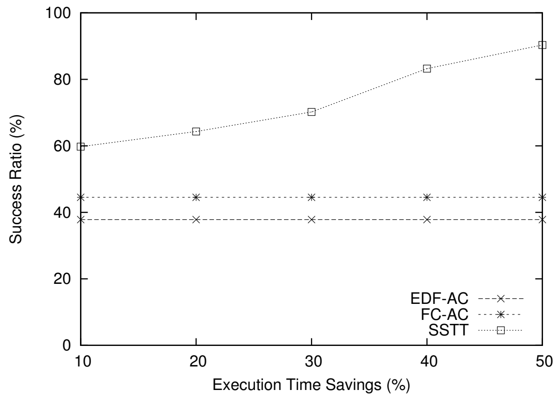

Figure 8. ETS vs. Success Ratio

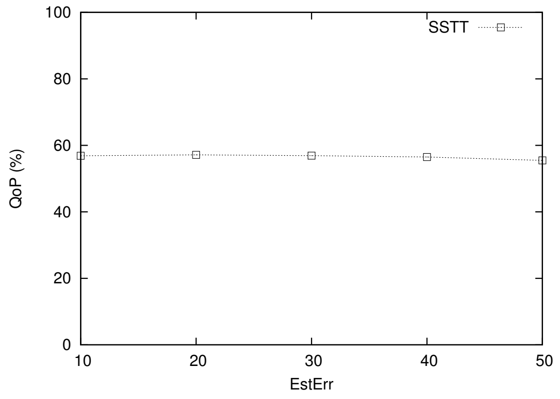

Figure 9. ETS vs. QoP

As shown in Figure 8, the success ratio of our approach increases as the ETS increase from 10% to 50%. Note that the success ratio of SSTT is approximately 90% when ETS = 50%, while EDF-AC and FC-AC show the success ratio lower than 50%. Thus, the success ratio of SSTT increases by approximately 30% when the ETS increases from 10% to 50%. At the same time, the QoP of SSTT ranges between $55\% - 58\%$ as shown in Figure 9.

Figure 10 shows the transient performance of SSTT when ETS = 20%. The patterns of the transient utilization and QoP variations are similar to Figure 7; however, SSTT

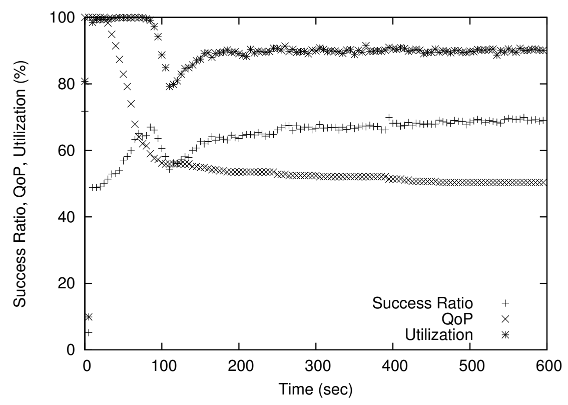

Figure 10. Transient Performance of SSTT (ETS = 20%)

shows the higher success ratio in Figure 10 than it does in Figure 7 due to the higher ETS. The transient success ratio of SSTT continuously increases as the ETS increases beyond 20%, similar to the average success ratio shown in Figure 8. However, we do not include all the transient performance results due to space limitations.

In resource constrained RTESs, it is highly likely for cryptographic functions to consume a large portion of the execution time of a secure real-time task. Therefore, our approach could considerably improve the success ratio by adapting the QoP under transient overload in a reliable, efficient manner.

5 Related Work

Our QoP adaptation is inspired by the milestone approach [8] in which the QoS is monotonically improved as a radar image, for example, is processed more. Similarly, in our approach, a longer key improves the QoP at the cost of the increased execution time. The main difference is that our approach quantifies the degree of QoP degradation in terms of key length and the time period during which the key length decrease is acceptable.

Cryptographic security support has rarely been studied in the context of real-time systems. QRAM [7, 17] selects an appropriate encryption key length based on the importance of an application and its resource requirements. However, QRAM requires *a priori* knowledge of execution times to optimize the QoS. Further, the selection of the key length only occurs at the start of an application, e.g, a video conference, unlike our approach.

Son et al. [20] developed an adaptive security manager in a real-time database. When the real-time database is overloaded, a weaker encryption algorithm is used to improve the deadline miss ratio. However, their approach does not quantify the degree of QoP degradation unlike our approach. Neither does it derive the time period in which a weaker encryption algorithm can be used safely. In addition, storing several keys rather than several encryption algorithms may require less memory in real-time embedded systems with limited resources.

Although the notion of Quality of Protection has been introduced in [9] to integrate the security and QoS support, it is not clearly known yet how to measure the quality of general security service. In this paper, we suggest to use the key length as a QoP metric in the context of real-time embedded systems. Spyropoulou et al. [21] have proposed the notion of QoSS (Quality of Security Service). Ideally, a system administrator and a security officer can select an appropriate security scheme to optimize the cost-benefit relation, when a quantitative model showing the computational cost and benefit of a security service is given. However, they give no specific model that can be used for the cost-benefit analysis. Also, they do not consider real-time constraints. In general, the cost-benefit analysis of security services is an open problem. We have taken a first step to solving this problem focused on systematic cryptographic security and timeliness trade-offs in real-time embedded systems.

Miyoshi et al. [14] have developed a novel access control scheme using the resource control lists to protect time-multiplexed resources such as the CPU and network bandwidth against some DoS (Denial of Service) attacks. Their work is complementary to our work. For example, we can use the resource control lists to protect real-time embedded systems against some DoS attacks, while balancing timing and cryptographic security requirements.

The access control problem in the multilevel security model has been studied in the real-time database literature [3, 19, 20, 6]. A majority of these work [3, 19, 20] temporarily allow a covert channel, which can be used by an adversary to enable an illegal information flow between different security levels, to improve the timeliness under overload conditions. George et al. [6] propose a secure real-time concurrency control protocol to avoid a covert channel. However, none of these work considers issues related to cryptographic security support.

Feedback-based real-time scheduling and QoS management have recently attracted a lot of attention since the several initial publications [1, 2, 12]. Our feedback control approach is similar to [12, 13]. Specifically, we make the utilization control model [12, 13] more formal by applying the system identification technique [16]. Further, security issues are not considered in [12, 13]. In fact, our adaptive QoP management policy is independent of the underlying scheduling algorithm. In this paper, however, we aim to show that our QoP management approach can be seamlessly integrated with advanced feedback control techniques to safely improve the success ratio even in the presence of dynamic workloads. As a result, SSTT can considerably improve the success ratio compared to the baselines including the feedback-based utilization control as shown in Section 4.

6 Conclusions and Future Work

A number of real-time embedded systems are employed in important applications, e.g., electric grid management or defense applications. In these systems, it is essential to meet deadlines, while supporting the confidentiality, integrity, and authenticity of messages. Despite the importance, cryptographic security support in real-time embedded systems has rarely been explored. To address this problem, we propose a systematic security and timeliness tradeoff policy integrated with a feedback control scheme. Our approach for security adaptation is different from existing work in that it can quantitatively measure the QoP degradation, while regulating how long a short key can be used to handle transient overloads. In the simulation study, our approach significantly improves the success ratio, at the cost of controlled security adaptation, especially when the system is suffering transient overloads paying relatively high computational costs for the cryptographic security support considered in this paper. Therefore, we have observed that our approach can efficiently support essential cryptographic security features in resource constrained real-time embedded systems. In the future, we will further investigate efficient cryptographic security support in real-time embedded systems. In addition, we will investigate other security issues related to real-time embedded systems, e.g., detection of (distributed) denial of service attacks.

References

[1] T. F. Abdelzaher and N. Bhatti. Adaptive Content Delivery for Web Server QoS. In *International Workshop on Quality of Service*, June 1999.

[2] T. F. Abdelzaher and K. G. Shin. End-host Architecture for QoS-Adaptive Communication. In *IEEE Real-Time Technology and Applications Symposium*, June 1998.

[3] Q. Ahmed and S. Vrbsky. Maintaining Security in Firm Real-Time Database Systems. In *14th Annual Computer Security Applications Conference*, 1998.

[4] M. Blaze, W. Diffe, R. L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security, A Report by An Ad Hoc Group of Cryptographers and Computer Scientists, January 1996. Available at http://www.crypto.com/papers/.

[5] J. Deamen and V. Rijmen. Efficient Block Ciphers for Smartcards. In *USENIX Workshop on Smartcard Technology*, 1999.

[6] B. George and J. R. Haritsa. Secure Concurrency Control in Firm Real-Time Databases. *Distributed and Parallel Databases*, 5:275–320, 1997.

[7] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek. On Quality of Service Optimization with Discrete QoS Options. In *the 4th IEEE Real-Time Technology and Applications Symposium*, 1998.

[8] K. J. Lin, S. Natarajan, and J. W. S. Liu. Imprecise Results: Utilizing Partial Computations in Real-Time Systems. In *Real-Time System Symposium*, December 1987.

[9] J. Linn. Generic Security Service Application Program Interface. IETF Request for Comments: 1508, 1993.

[10] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[11] J. Loyall, R. Schantz, D. Corman, and J. P. andS. Fernandez. A Distributed Real-Time Embedded Application for Surveillance, Detection, and Tracking of Time Critical Targets. In *The 11th IEEE Real-Time Embedded Technology and Applications Symposium*, 2005.

[12] C. Lu, J. Stankovic, G. Tao, and S. Son. Design and Evaluation of a Feedback Control EDF Algorithm. In *Real-Time Systems Symposium*, December 1999.

[13] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms. *Real-Time Systems, Special Issue on Control-Theoretical Approaches to Real-Time Computing*, 23(1/2), May 2002.

[14] A. Miyoshi and R. Rajkumar. Protecting Resources with Resource Control Lists. In *IEEE Real Time Technology and Applications Symposium*, 2001.

[15] National Institute of Standards and Technology, Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard, Nov. 2001.

[16] C. L. Phillips and H. T. Nagle. *Digital Control System Analysis and Design (3rd edition)*. Prentice Hall, 1995.

[17] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. Practical Solution for QoS-based Resource Allocation Problems. In *IEEE Real-Time Systems Symposium*, December 1998.

[18] B. Schneier. *Applied Cryptography*. Wiley, 2nd edition, 1996.

[19] S. H. Son, R. Mukkamala, and R. David. Integrating Security and Real-Time Requirements using Covert Channel Capacity. *IEEE Transactions on Knowledge and Data Engineering*, 12(6), Dec 2000.

[20] S. H. Son, R. Zimmerman, and J. Hansson. An Adaptable Security Manager for Real-Time Transactions. In *Euromicro Conference on Real-Time Systems*, pages 63–70, Stockholm, Sweden, June 2000.

[21] E. Spyropoulou, T. Levin, and C. Irvine. Calculating Costs for Quality of Security Service. In *15th Computer Security Applications Conference*, 2000.

[22] M. J. Wiener. Efficient DES Key Search. Technical Report TR-244, Carleton University, May 1994.