# Dynamic Security and QoS Adaptation 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

*A number of real-time embedded systems (RTESs) are used to manage critical infrastructure such as electric grids or $C^4I$ systems. In these systems, it is essential to meet deadlines, for example, to avoid a power outage or loss of a life. The importance of security support is also increasing, because more RTESs are being networked. To securely transmit sensitive data, e.g., a battle field status, across the network, RTESs need to protect the data via cryptographic techniques. However, security support may cause deadline misses or unacceptable QoS degradation. As an initial effort to address this problem, we formulate the security support in RTESs as a QoS optimization problem. Also, we propose a novel adaptive approach for security support in which a RTES initially uses a relatively short cryptographic key to maximize the QoS, while increasing the key length when the security risk level is raised. In this way, we can make a possible cryptanalysis several orders of magnitude harder by requiring the attacker to search a larger key space, while meeting all deadlines by degrading the QoS in a controlled manner. To minimize the overhead, we derive the appropriate QoS levels for several key lengths via an offline polynomial time algorithm. When the risk level is raised online, a real-time task can use a longer key and adapt to the corresponding QoS level (derived offline) in O(1) time.*

## 1 Introduction

A number of real-time embedded systems (RTESs) are used to manage critical infrastructure such as electric grids and $C^4$I (Command, Control, Communications, Computers, and Intelligence) systems. In these systems, meeting deadlines is essential, since a deadline miss may cause a catastrophic result, e.g., a power outage or loss of a life. The importance of security support is also increasing, since more RTESs, e.g., $C^4$I systems, are being networked. Although cryptographic security and real-time systems have been well studied *separately*, very little work has been done to meet both timing and cryptographic security requirements.

Simultaneously supporting security and timing constraints is challenging, since these requirements can compete for computational resources. A simplistic approach, e.g., always using the longest key, may incur severe QoS degradation in resource constrained RTESs, often based on cheap, low-end microprocessors and/or microcontrollers. (It is known that the time for encryption increases as the key length increases [5].) In addition, sensor and control data have to be protected for a relatively short time, since they change frequently. Unless the current risk is high, too hard an armor could be excessive, wasting precious resources. Thus, we aim to handle potentially time-varying risks in a cost-effective manner.

In this paper, we first formulate the security support in RTESs as a QoS optimization problem. In addition, we present a novel *adaptive* and *risk-aware* approach to efficiently satisfying security and timing constraints in resource constrained RTESs. When a network intrusion detection system raises the level of the security risk, e.g., due to the detection of attempts for illegal data access, a RTES begins to use a longer cryptographic key to better protect sensitive data. As a result, a possible cryptanalysis trying to find the cryptographic key is forced to take several orders of magnitude longer. At the same time, the RTES may have to degrade the QoS of certain real-time tasks to avoid potential deadline misses due to the increased security cost.

To minimize the adaptation overhead, we perform an offline analysis to optimize the QoS for several different key lengths. In this way, we ensure all deadlines will be met if the total utilization is smaller than or equal to the schedulable utilization bound, e.g., 1 in EDF, while maximizing the QoS for given key lengths. Given the current risk level, our approach can dynamically select the appropriate key length and QoS levels of real-time tasks from the precomputed settings in O(1) time. Consequently, our approach can support desirable security and real-time features with minimal overheads.

The remainder of the paper is organized as follows. The scope of the work is described by discussing the basic assumptions and formulating the problem of security and real-time performance management in Section 2. Our online and offline algorithms for security and timing assurance are presented in Section 3. Related work is discussed in Section 4. Finally, Section 5 concludes the paper and discusses the future work.

## 2 Scope of the Work

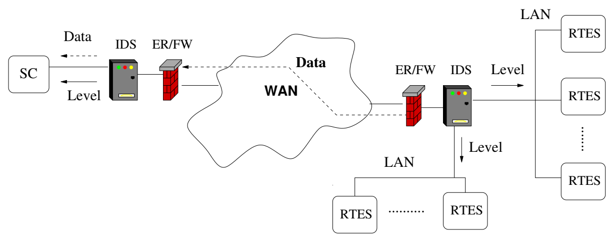

**Figure 1. System Model: Example**

**Application Scenario and System Model.** Figure 1 shows a high level diagram of example RTESs and a service center (SC) that operate as follows. A RTES monitors and controls, e.g., (part of) an electric grid or a battle field. A RTES may transmit sensor or control data to a SC to report, e.g., an overload of a power generator. RTESs in a specific area are connected via a local area network (LAN) such as a control area network (CAN), while communicating with a SC across the wide area network (WAN) through an edge router (ER) that connects the LAN to the backbone network.

We assume that each task in a RTES is associated with a hard deadline; however, the transmission delay between a RTES and SC does not require a hard guarantee. We also assume that RTESs are independent of each other to avoid a deadline miss due to an unpredictable delay for communication. We assume that the communication medium access, transmission, and propagation delay in the LAN, e.g., CAN, is predictable and the delay is included in the schedulability analysis to meet all deadlines, similar to [6]. We also assume a RTES transmits data in asynchronous mode, in which control is immediately returned to a real-time task without waiting for the completion of the transmission.

An ER is equipped with a firewall (FW) to filter suspicious packets and connection requests, e.g., insecure file transfer protocol (FTP) connection requests. Note that a firewall does not completely solve the network security problem, because an adversary, who has access to the wide area network, may eavesdrop or inject packets to the network. For example, a terrorist may try to find the weakest point of an electric grid by eavesdropping. Or, he may attempt to send false data to the SC. Hence, it is necessary to encrypt and authenticate data to support the confidentiality, integrity, and authenticity of the network messages.

An intrusion detection system (IDS) monitors incoming and outgoing packets and periodically broadcasts the current risk level to the connected RTESs or a SC in a specific area. (We assume the intrusion detection and broadcast task is sporadic and can subsequently be mapped as a periodic task. A thorough investigation is reserved for future work.) A SC is in charge of a specified set of RTESs. It takes care of, e.g., long-term plans for energy supply and military tactics, based on the information received from RTESs (and other sources, if any). We assume that a SC is also protected via a firewall and IDS, similar to RTESs.

**Threat Model.** We assume that an isolated RTES is trusted. Thus, a RTES only has to encrypt data before transmitting them across the network. We also assume that an IDS is not compromised.

We consider a symmetric key system in which a SC 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 [5]. We support the confidentiality, integrity, authenticity of a message via encryption and cryptographic one-way hash function. Therefore, an adversary cannot eavesdrop, corrupt, or inject messages without being detected. In addition, we support semantic security and replay protection. For more details, refer to [1].

Each RTES shares a pair of secret keys with a SC to encrypt and authenticate a message to support the message confidentiality, integrity, and authenticity. We use different keys for encryption and authentication, since the rule of thumb is to use separate keys for different applications [5]. Note that only the SC can decrypt and check the integrity and authenticity of the message transmitted by the RTES using the shared keys.

Finally, it is assumed that main attacks against a scrutinized cryptosystem without known vulnerability, e.g., AES (Advanced Encryption Standard) [5], are *brute-force attacks* that tries to find the secret key via an exhaustive search in the key space. Note that this is a common assumption in trusted cryptosystems [5].

**Problem Formulation.** In this paper, we assume that a RTES consists of a set of $N$ periodic tasks $< T_1, T_2, ..., T_N >$. We assume the deadline of a task is equal to its period. The (worst case) execution time of a task $T_i$ is: $C_i = C_{i,c} + C_{i,e(l)}$ where $C_{i,c}$ is the real-time function execution time and $C_{i,e(l)}$ is the data encryption (and transmission) time for the key length $l$, respectively. The utilization of a task $T_i$ is: $U_i = C_i/P_i$ where $P_i$ is the period of $T_i$. Further, each task is associated with $k(\geq 1)$ discrete QoS levels. The QoS monotonically increases as the QoS level (and the corresponding execution time) increases, similar to the milestone approach [3]. Ideally, we aim to optimize the aggregate QoS of the real-time tasks $\{T_1, T_2, ..., T_N\}$ defined in terms of both security and real-time performance, while addressing the current security risk indicated by an IDS:

$$\text{Maximize } QoS = S(l) \cdot \frac{\sum_{i=1}^{N} Q(i)}{\sum_{i=1}^{N} max\ Q(i)} \leq 1 \qquad (1)$$

where $Q(i)$ and $max\ Q(i)$ are the current and maximum possible QoS of a task $T_i$ and

$$S(l) = \begin{cases} 1 \text{ if } S(l) \geq R(t) \\ 0 \text{ otherwise} \end{cases} \qquad (2)$$

where $S(l)$ indicates the strength level of the currently used cryptosystem, which is determined by the current key length $l$ measured in number of bits. For example, a security officer can set $S(128) = 1, S(192) = 2$, and $S(256) = 3$. $R(t)$ in Eq 2 denotes the current risk level estimated by the IDS, e.g., 1 (low), 2 (medium), and 3 (high). Specifically, we select the key lengths to defend the cryptosystem against a potential cryptanalysis for a certain estimated period of time. (Refer to [1] for more details.) Also, note that our QoS optimization is subject to

$$\sum_{i=1}^{N} U_i \leq UB \qquad (3)$$

where the $UB$ is the utilization bound of an employed real-time scheduling algorithm such as EDF. When the risk level $R(t)$ is raised by the IDS, a task $T_i$ has to use a longer key. As a result, $C_i$ and $U_i$ may increase, and therefore, the QoS manager should reallocate resources to re-optimize the QoS. Otherwise, the QoS may become suboptimal, possibly incurring deadlines misses if Eq 3 cannot be satisfied.

## 3 Security and Timeliness Assurance

Our online security and QoS adaptation algorithm is summarized in the following.

1. Periodically receive the current risk level $R(t)$ at time $t$ from the IDS.
2. If the current strength of defense $S(l) < R(t)$, use a new key that is $l'$ bits long where $S(l') = R(t)$ and $l' > l$.
3. Degrade the QoS of certain real-time tasks, if necessary, to meet all deadlines.

Our offline precomputation algorithm for a specific key length is summarized and discussed as follows.

```
1. Input: Key length $l$ and QoS function $g_i : u_i \rightarrow q_i$ for each task $T_i$ where $1 \leq i \leq N$ (#tasks in the system) and $u_i$ and $q_i$ are the utilization and corresponding QoS, respectively.
2. Output: $Q\_List[l][i]$
3. for $(i = 1;\ i \leq N;\ i++)$
    For a task $T_i$, derive the convex hull $H_i = \{<u_{i1}, q_{i1}>, ..., <u_{ij}, q_{ij}>\}$ where $j \leq L$ (#max QoS levels).
4. /* Merge sort in nondecreasing order of utilization */
    $H\_List = Merge(H_1 - \{<u_{11}, q_{11}>\}, H_2 - \{<u_{21}, q_{21}>\}, ..., H_N - \{<u_{N1}, q_{N1}>\})$
5. $Q\_List[l] = \emptyset$ (empty set)
6. $U^a = UB$
7. /* Support at least the minimum QoS */
for $(i = 1;\ i \leq N;\ i++)$
{
  a. if $(U^a < u_{i1})$
  b.   print("Error: Cannot Support Key Length $l$")
  c.   exit(-1)
  d. $Q\_List[l][i] = q_{i1}$
  e. $U^a = U^a - u_{i1}$
}
8. /* Initial resource allocation */
for $(i = 1;\ i \leq N;\ i++)$    $U[i] = u_{i1}$
9. /* QoS optimization */
for $(i = 1;\ i \leq |H\_List|;\ i++)$
{
  a. $id = H\_List[i].id$
  b. $\delta U = H\_List[i].u - U[id]$
  c. if $(\delta U > U^a)$ exit(0)
  d. $Q\_List[l][id] = H\_List[i].q$
  e. $U^a = U^a - \delta U$
  f. $U[id] = H\_List[i].u$
}
```

In our approach, we assume a RTES stores the secret keys in an array $key[1..|R|]$ where $|R|$ is the number of risk levels considered by the IDS. For each key length, as shown in Steps 1 and 2, we precompute the near optimal QoS levels of the real-time tasks offline to minimize the overhead for online security and QoS adaptation, if necessary. For a specific key length $l$, task $T_i$ can simply switch to the QoS level stored in $Q\_List[l][i]$ when the system needs to use a key of length $l$ to ensure that the level of defense is higher than or equal to the risk level.

In Step 3, as shown in Figure 2, we remove suboptimal discrete QoS levels by deriving the convex hull for each task $T_i (1 \leq i \leq N)$, similar to [2]. In this way, a list of near optimal $< utilization, QoS >$ pairs, i.e., $\{<u_{i1}, q_{i1}>, <u_{i2}, q_{i2}>, ..., <u_{ij}, q_{ij}>\}$, can be derived for each task $T_i$ in increasing order of utilization where $u_{ij}$ is the utilization required to support task $T_i$'s QoS level $j$.

In Steps $4 - 7$, we initialize $Q\_List[l]$ to support at least the minimum required QoS when the key is $l$ bits long. In Step 4, all the $< utilization, QoS >$ pairs of all tasks except $<u_{i1}, q_{i1}>$ are merged in nondecreasing order of utilization. If two pairs have the same utilization, the one with the higher QoS is added to the list first. A tie, in which two pairs have the same utilization and QoS, is broken in a random manner. In Step 6, the available utilization $U^a$ is initialized. In Steps 7.a−7.c, we check whether $U_a$ is enough to support the deadline and the minimum QoS of $T_i$. If this test fails, the RTES cannot support the specific key length $l$ and the algorithm terminates with an error. If the required utilization $u_{i1}$ to support the minimum QoS of $T_i$ is available, we set $Q\_List[l][i] = q_{i1}$ and subtract $u_{i1}$ from $U^a$ in Steps 7.d and 7.e. The procedure is repeated for the next task $T_{i+1}$, if any.

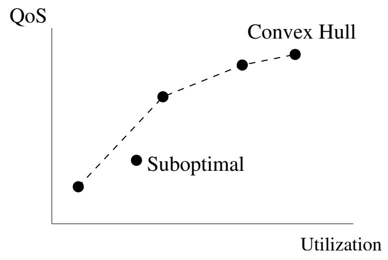

**Figure 2. Example QoS Function**

In Step 8, the required minimum utilization of each task $T_i$ $(1 \leq i \leq N)$ needed to support at least $T_i$'s minimum QoS is initialized. In Step 9.a, we remove the first task in the $H\_List$, i.e., $T_{id}$, which requires the smallest utilization. $\delta U$, i.e., the extra utilization needed to improve the QoS of $T_{id}$, is computed in Step 9.b. If $\delta U > U^a$, the algorithm terminates normally in Step 9.c. (We can allocate remaining resources in a greedy manner; however, we do not take the greedy approach to reduce possibly excessive QoS fluctuations when the risk level changes.) Otherwise, the QoS of $T_{id}$ is raised to the next higher level in Step 9.d. In Steps 9.e and 9.f, the available utilization $U^a$ and the utilization allocated to $T_{id}$ are updated, respectively. The whole procedure is repeated for every key length used by a RTES. Consequently, the algorithm generates a table $Q\_List[1..\#keys][1..\#tasks]$ that can be used by our online algorithm to select an appropriate key length and QoS levels considering the current risk level in O(1) time. Finally, the time complexity of our offline algorithm is $O(KLNlogLN)$, which becomes $O(NlogN)$ when the number of keys $(K)$ and QoS levels $(L)$ are constant.

## 4 Related Work

Cryptographic security support has rarely been studied in the context of real-time systems. Lee et al. [2] propose to select an appropriate encryption key length based on the importance of an application and its resource requirements. However, the selection of the key length and QoS levels only occurs at the start of an application, e.g, a video conference, unlike our approach. Son et al. [7] 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 consider the current risk level, if necessary, to strengthen security. Although the notion of Quality of Protection has been introduced in [4] 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 a QoS function defining the quality of security and real-time services and security/QoS trade-off algorithms.

## 5 Conclusions and Future Work

A number of real-time embedded systems are used to manage critical infrastructure, e.g., electric grids or $C^4$I systems. In these systems, it is essential to meet deadlines, while supporting the confidentiality, integrity, and authenticity of network messages. Despite the importance, security support in RTESs has rarely been supported. To address this problem, we formulate the problem as a QoS optimization problem and propose offline and online algorithms for security and QoS adaptation to address dynamic security risks, while optimizing the QoS. In this way, our approach efficiently supports essential cryptographic security features in resource constrained RTESs. In the future, we will further investigate efficient cryptographic support in RTESs considering different task models such as sporadic or aperiodic models. Further, we will investigate other security issues closely related to RTESs, e.g., detection of (distributed) denial of service attacks.

## References

[1] K. Kang. On Efficient Cryptographic Security Support in Real-Time Embedded Systems. Technical Report CS-TR-05-KD02, State University of New York at Binghamton, Oct. 2005. Available at http://www.cs.binghamton.edu/∼kang.

[2] 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.

[3] 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.

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

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

[6] J. W. S.Liu. *Real-Time Systems*. Prentice Hall, 2000.

[7] 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.