Secure Software Attestation for Military Telesurgical Robot Systems

Kyle Coble
SIS Dept. and CyberDNA
UNC Charlotte
kjcoble@uncc.edu

Weichao Wang
SIS Dept. and CyberDNA
UNC Charlotte
weichaowang@uncc.edu

Bill Chu
SIS Dept. and CyberDNA
UNC Charlotte
billchu@uncc.edu

Zhiwei Li
SIS Dept. and CyberDNA
UNC Charlotte
zli19@uncc.edu

***Abstract*—Telesurgical robot systems (TRS) are often deployed in unattended environments such as battlefields or rural areas. Therefore, adversaries can easily access the devices, compromise the system, and install their own malware. If the integrity and health of the system software and configuration files are not verified before their usage, the safety and lives of the injured soldiers and patients may be in danger. Many existing software attestation mechanisms depend on the calculation delay to distinguish a correct memory image from a compromised system. We cannot directly apply this technique to transcontinental TRS when we consider the long transmission delay between the verifier and the prover. In this paper, we propose a software attestation mechanism that can distinguish between these two kinds of delay. A secure communication protocol among the verifier, telesurgical robot, and secure token of the remote medical personnel is designed. The safety of the approach is analyzed and its overhead is evaluated.**

I. Introduction

Telesurgical Robot Systems (TRS) have a significant potential in military operations and disaster relief operations by providing real-time assistance to the medical personnel when the necessary expertise is not available on site. For example, an initiative led by the Telemedicine Advanced Technical Research Center (TATRC) at U.S. Army Medical Research & Materiel Command (USAMRMC) has allowed a specialist at Brooke Army Medical Center, Texas to provide real-time advice and instructions through the Medical Communications for Combat Casualty Care (MC4) systems so that a doctor at the 47th Combat Support Hospital can accomplish a complex surgical procedure [1]. The development and deployment of this technique becomes an essential component in the future war against terror and other foreign military operations.

Because of the complexity of the systems, investigators have focused their research on the development of the latest techniques in mechanical, electrical, and robotics and the construction of the prototype systems. Not enough attention has been paid to the security of the systems. However, considering the hostile environments such as a battlefield in which the TRS will be deployed, we must carefully address the challenges in security, safety, and robustness [2]. Previous research on the security of the telesurgical robot systems focuses on the problems such as user authentication and access control. For example, the SecureITP protocol [3] uses the certificates based on X.509 to authenticate the surgical robots and uses AES encryption to protect the data traffic. The EUROMED-ETS approach [4] uses trusted third parties to manage the public keys of the medical practitioners and to enforce access control. Abuse case models [5] are established to investigate the end-to-end security in telemedical networks. The security of a simulated kyphoplasty telesurgery system is studied in [6]. While these approaches solve some problems in securing the telesurgical robot systems, the security issues other than encryption and authentication have not been extensively studied.

In this paper, we plan to investigate the remote software attestation of the telesurgical systems. Under many conditions, the TRS systems are deployed in an unattended manner. Therefore, adversaries can easily access the devices, compromise the system, and install their own malware. If the integrity and health of the system software and configuration files cannot be verified before their usage, the safety and lives of the injured soldiers and patients may be jeopardized. Current remote software attestation approaches rely on filling the empty memory with pseudo random "noise", traversing the program memory in pseudo-random patterns, and timing the attestation procedure to verify the integrity of the OS. For example, in [7], [8], [9], if the correct attestation response is returned within a time window, the device is considered safe. The extra calculation delay introduced by the malicious OS can be detected by the verifier.

Many software attestation approaches assume a direct, short-range wireless connection between the verifier and prover so that the transmission delay between them can be ignored. This assumption, however, does not hold any more in a military, transcontinental telesurgical robot system. For example, a TRS deployed in Iraq needs to transmit the information through a multi-hop path consisting of UAV (Unmanned Aerial Vehicle), satellite, and local area networks to reach the medical center in Texas. The one-way transmission delay can be as long as 500 to 1000 milliseconds [10]. Therefore, the malicious attackers can easily hide the extra computation time during the attestation under the transmission delay and deceive the verifier. Note that the digital signatures of the TRS over the self-measured attestation duration will not solve the problem when the device has been compromised by the attacker.

To solve this problem, in this paper we propose an attestation procedure that uses a smart card to perform the time measurement for the verification. This card will only draw power from the TRS, and it has its own processing unit.

978-1-4244-8179-8/10/$26.00 ©2010 IEEE

When an attestation is needed, the remote verifier will send the parameters for the attestation to the smart card through the TRS. The smart card will then provide the parameters to the TRS and measure the computation time. Finally, the smart card will calculate the measurement results, generate a digital signature, and transmit the results back to the verifier. The verifier can then examine the calculation delay and the attestation result to diagnose the health of the system.

Using a smart card to measure the attestation duration provides a unique advantage over traditional approaches. Through separating the device under attestation from the time keeper, we ensure that a compromised TRS cannot hide behind network latency. This is a crucial step in verifying that these systems are safe and stable before relying on them to perform sensitive tasks. To demonstrate the feasibility of our approach, we will analyze the overhead and the safety of the proposed mechanism. Our approach will ensure that telesurgical robot systems can be attested and verified to be safe, regardless of their locations or their network connection speed.

The remainder of this paper is organized as follows. In Section II, we present the details of our approach to resolving the remote attestation problem of a TRS. Section III provides an analysis of the performance and safety of our approach. In Section IV, we review the related work. Finally, Section V summarizes our work and explains how we plan to further refine the design.

II. Proposed Approach

A. System Assumptions

An abstract model of the software attestation problem can be described as follows. We assume that a control component (the verifier) that is far from the TRS (the prover) needs to remotely verify the integrity of the system software. We assume that the verifier knows the expected program memory contents at the prover. This assumption is reasonable for critical real-time systems such as a TRS for a number of reasons.

First, the OS for TRS is usually a specially designed system that does not perform any other tasks. The updates to the TRS system (such as a software patch) can be issued only by the remote control component. Second, to guarantee the cleanliness of the system, under many conditions the remote verifier will force the TRS to reboot before performing an important task. In this way, the verifier will have an accurate depiction of the program memory contents of the TRS.

We assume that the TRS contains a memory-content-verification function that the verifier can remotely activate. Here we do not require the verification function to be stored on a tamper-proof hardware and the attackers can change the function freely if they compromise the TRS. However, since the verifier has the same function, the final attestation results will be different if the attackers have changed the verification function. We assume that the verifier and prover share a secure, lightweight pseudo random number generator (PRNG) [11]. In this way, the two nodes can generate the same memory access patterns when they have the same seed for the PRNG.

The TRS will contain a smart card slot and medical personnel have to insert their smart card to activate the TRS. The smart card has its own processing unit that can perform the operations such as digital signature [12]. The smart card can also measure a time duration at the accuracy level of $10^{-6} second$. This can be achieved since the GPS chip set has already been embedded into the SIM card and smart card [13]. Using its public/private key pair, the smart card can communicate securely with the verifier through the prover. The prover can discard the packets between the smart card and the verifier if it has been compromised. However, it does not have the computation power to directly compromise the encryption key.

We assume that the attackers will have full control over the TRS if it is compromised. This means they can install malware on the system and copy the original OS to other parts of program memory or the hard-drive. However, the attackers cannot change the hardware architecture of the TRS (e.g. add another 1GB memory, boost the CPU speed, *etc*). This can be enforced by implementing the TRS on a specially designed single board computer, which is a popular solution for many telesurgical robot systems [14].

B. Overview of Current Approaches

In this part we describe the current approaches to software attestation [7], [8], [9] and explain why they cannot be directly applied to a TRS. As shown in Figure 1, the prover has the attestation function in its program memory. When the verifier sends the attestation request with a random seed for PRNG and a nonce to the prover, the prover will activate the PRNG with the seed to generate the pseudo random memory access sequence. The prover will then concatenate the nonce and the accessed program memory contents and use a secure hash function to generate the attestation result as $hash(nonce, accessed\ value\ 1, accessed\ value\ 2, \cdots, accessed\ value\ n, nonce)$. The final calculation result will be sent back to the verifier. The verifier will use its own copy of the expected program memory to calculate the hash result and compare the two values. If the two values are the same, the integrity of the TRS is verified. Note that the seed and the nonce values will prevent an attacker from conducting the pre-computation attacks. To prevent the attacker from using the free program memory space to store the malicious code, the attestation function will generate pseudo random numbers to occupy the free spaces [8], [9].

At the bottom of Figure 1, the attackers compromise the TRS and the malicious OS controls the device. Since the pseudo random numbers in the free space of the program memory will also participate in the hash calculation, the attackers will copy the contents of the original OS from the occupied area to the data memory or the hard drive. Note that the attestation function can still execute under the control of the malicious OS. However, during the random memory access procedure, an 'if' operation will be inserted before every 'load' operation to determine whether or not the accessed memory has been moved by the attacker. Since the hash calculation is

very efficient, the conditional jump caused by the 'if' operation will greatly increase the execution time of the attestation function. For example, each round of the main loop of the attestation function in [7] needs only 23 CPU cycles, while the 'if' operation needs 3 cycles. When the attestation function reads hundreds of thousands of memory locations, the increase in execution time will reach several hundred milliseconds. If the verifier and the prover have a direct wireless connection, this increase can be easily detected. Therefore, even if the prover correctly calculates the hash result, the verifier can still detect the compromise through the time delay.

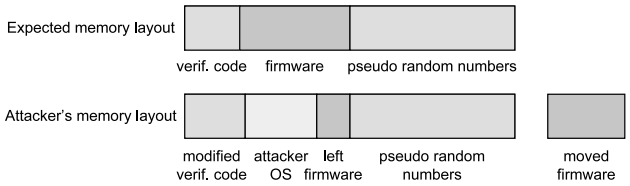

Fig. 1. Current software attestation approaches.

There is a major drawback when we have to measure the attestation duration over a multi-hop, long distance connection between the verifier and the TRS. The one-way transmission delay between a telesurgical system in Iraq and the medical center in US can easily reach 800 to 1000 milliseconds, which will allow an adversary to execute the modified code, return a "correct result", and hide the extra calculation time in the transmission delay. The digital signature generated by the TRS over the measured attestation duration cannot be used to defend against such attacks since the TRS has already been compromised. Therefore, a new mechanism must be designed to solve this problem.

C. Proposed Attestation Protocol

Based on the analysis in Section II.B, we find that the major reason current attestation mechanisms cannot be applied to telesurgical systems is because we cannot distinguish the network delay from the attestation calculation time. When we reexamine the application scenario of the TRS, we find that when a telesurgery is being conducted, one or several medical personnel are usually present at the remote site. Although they may not have any computer security knowledge, their identity tokens such as the smart card can help the verifier perform the attestation function.

The application scenario of the proposed attestation protocol is illustrated in Figure 2. We assume that every medical personnel has a smart card to prove her/his identity. The smart card has its own public/private key pair and can conduct asymmetric encryption/decryption. The smart card also contains a GPS chip set on it so that it can receive the accurate time provided by the satellites. When a smart card is inserted into the TRS, it can communicate with the robot without any delay. At the same time, the TRS will serve as the router between the verifier and the smart card.

The basic idea of the proposed attestation protocol is as follows. When the smart card is inserted into the telesurgical system, it sends an attestation request to the verifier. The verifier will generate the seed for the random memory access patterns and the nonce for hash calculation. It will encrypt the values with the public key of the smart card and deliver the message to the smart card with the help of the TRS. When the smart card recovers the seed and nonce, it will provide the numbers to TRS and start its clock. The TRS will execute the attestation function and send the final result back to the smart card. The smart card will then generate the digital signature on the hash result and the measured time duration and send the values to the verifier so that it can compare the result to its own calculation.

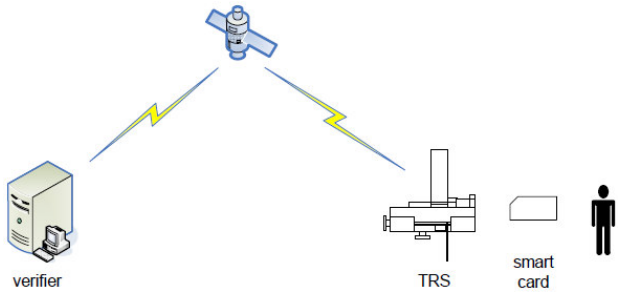

Fig. 2. Smart card assisted software attestation of the TRS.

During this procedure, the measurement of the attestation calculation time is conducted by the smart card. Although the seed for the random memory access patterns and the nonce for hash calculation are forwarded to the smart card by the TRS, the asymmetric encryption with the public key of the smart card prevents the TRS from gaining access to these values and conducting pre-computation. The smart card will measure the time duration between the seed and nonce are provided to the prover and the hash result is returned. Since the smart card and the TRS have a direct communication channel, we can effectively separate the network transmission delay from the attestation time.

$$c \rightarrow p : m_1 = (cert_c, E_{pri(c)}\{ID_c, r_c, timestamp\}) \quad (1)$$

$$p \rightarrow v : (m_1, cert_p, E_{pri(p)}\{ID_p, r_p, hash(m_1)\}) \quad (2)$$

$$v \rightarrow (p) \rightarrow c : (cert_v, E_{pub(c)}\{E_{pri(v)}\{ID_v, ID_c, r_v, r_c, E_{pub(p)}\{E_{pri(v)}\{ID_v, ID_p, r_v, r_p, seed, nonce\}\}\}\}) \quad (3)$$

$$c \rightarrow p : (cert_v, E_{pub(p)}\{E_{pri(v)}\{ID_v, ID_p, r_v, r_p, seed, nonce\}\}) \quad (4)$$

$$The\ TRS\ conducts\ the\ software\ attestation$$

$$p \rightarrow c : m_2 = (E_{pub(v)}\{E_{pri(p)}\{r_v, r_p, hash\ result\}\}) \quad (5)$$

$$c \rightarrow (p) \rightarrow v : (m_2, E_{pub(v)}\{E_{pri(c)}\{r_v, r_c, measured\ time\ duration, hash(m_2)\}\}) \quad (6)$$

Fig. 3. Details of the attestation protocol.

The details of the authentication and attestation procedures are illustrated in Figure 3. In these messages, we use the sub-index of 'v', 'p', and 'c' to represent the verifier, the prover (TRS), and the smart card, respectively. For example, $pub(c)$ and $pri(c)$ represent the public key and private key of the smart card. We use $E_{key}\{*\}$ to represent an asymmetric encryption. We assume that the verifier, the prover, and the smart card each have a valid certificate for their public keys issued by a trusted third party. The r_c, r_p, and r_v are random numbers

generated by the three parties to uniquely label the attestation procedure and defend against the replay attacks.

Below we provide an explanation of the protocol. In message (1) and (2), the smart card and the TRS send their certificates to the verifier to establish the secure communication channel. The $timestamp$ in message (1) and $hash(m_1)$ in message (2) are used to prove the freshness of the attestation request to the verifier. In message (3), the verifier sends the seed and nonce to the smart card through the TRS. Here the encryption with the public key $pub(c)$ of the smart card will prevent the TRS from getting access to the values. In message (4) and (5), the smart card provides the seed and nonce to the TRS and the TRS returns back the attestation hash result. Note that these messages are protected by the public keys of the TRS and verifier respectively. In this way, the smart card can only measure the attestation duration but cannot learn any information about the software system. Finally, in message (6), the smart card generates the digital signature of the measured time duration. It adds the hash value of m_2 into the digital signature to make sure that the time duration and the attestation result are bound together.

We use ProVerif [15], a widely used automated protocol analyzer, to verify the security of our attestation protocol under the standard Dolev-Yao intruder model. We assume that the cryptography algorithms are safe, and an intruder has the capability to overhear, intercept, and forge new messages. We are especially interested in the secrecy and integrity properties of the measured time duration. We use Horn clauses to encode the attestation protocol and the final verification results indicate that the attacker is unable to impersonate the smart card with a fake time duration.

The proposed approach has the following advantages:

• We successfully separate the measurement of the attestation duration from the communication delay. The direct connection between the smart card and the TRS will introduce a negligible delay. Therefore, existing attestation approaches can be applied to the TRS.

• An attacker has to compromise both the TRS and the smart card to allow an infected device to pass the software attestation and deceive the verifier. This will provide an extra layer of protection to the TRS. Detailed discussion on the safety of the approach will be presented in Section III.B.

• The approach uses only the operations that are supported by current smart card techniques. The attestation procedure is fully automated and transparent to the medical personnel at the remote site. Therefore, the adoption of the approach will not demand any extra training for the end users. The approach does not depend on any specific digital signature algorithm and it can evolve with the advances of smart card cryptography.

D. TRS for Users without Smart Cards

There are many emergency conditions on a battlefield and the TRS must adapt to such conditions. For example, we may need to perform telesurgery on an enemy soldier who will not have a smart card, or one of our soldiers who has lost her/his card. Under these conditions, the correctness of the attestation result cannot be guaranteed. We can adopt two methods to manage the allowed operations of the TRS under these conditions. In the first mechanism, we can classify the operations of the TRS into different groups based on their impacts on the patients. For example, we will allow the TRS to perform the operations with no or little impacts (e.g. basic measurements, sampling blood) even when the attestation cannot be conducted. On the other hand, the operations that will cause severe impacts on the patients (such as amputation) can be performed only after the system has successfully passed the attestation. In the second mechanism, the system administrator at the verifier side has the authority to remove any restrictions on the TRS that are set based on the attestation results. The motivation of this scheme is to hold the final decision making power in an intelligent person instead of a robot, which has been supported by the ethical evaluation of the impacts of the TRS on the patient-physician relationship [16].

III. Performance and Safety Analysis

In this Section, we evaluate the overhead of the proposed approach and investigate its safety.

A. Overhead of the Proposed Approach

The proposed approach incurs very little storage overhead for a TRS or smart card. At the same time, since the TRS needs to transmit a large amount of video traffic to the remote side once the surgery begins, the extra communication overhead caused by the protocol is negligible. Therefore, our analysis will focus on the computation overhead. The majority of the computation overhead is caused by generating and verifying the digital signatures. When we study the attestation protocol shown in Figure 3, we find that the TRS needs to conduct three public key encryptions and three decryptions, respectively. Since telesurgical systems typically operate on a CPU with high processing power, the encryption and decryption operations will not significantly impact the system performance. Note that when the smart card provides the seed and nonce to the TRS and when TRS sends back the attestation hash result, both messages are protected by two layers of asymmetric encryption. When the verifier examines the calculation time, this factor must be considered. Since the verifier knows the hardware architecture of the TRS, it can accurately estimate the required time to conduct these operations.

The smart card needs to conduct three asymmetric encryption and three decryption operations during the protocol. When we consider the limited memory space and processing power of the smart cards, these operations will incur a much longer computation time. Depending on the adopted digital signature algorithms and the card type, the computation time can be different. In Table I, we list the experiment results on the calculation time of several types of smart cards [17]. Based on the results, the overall attestation procedure will be shorter than 10 seconds, which is several degrees of magnitude shorter than the duration of a surgery. At the same time, all operations

of the smart card are autonomous and do not require any user interaction.

TABLE I
CALCULATION TIME OF ASYMMETRIC ENCRYPTION ON SMART CARDS.

Card type	Algorithm	Key length	Time
Hitachi H8/3113	RSA (sign)	512 bit	68 ms
Hitachi H8/3113	RSA (sign)	1024 bit	480 ms
Atmel MSC0501	RSA (sign)	1024 bit	1416 ms
SGS ST19KF16	RSA (sign)	512 bit	55 ms
SGS ST19KF16	RSA (sign)	1024 bit	380 ms
SGS ST19KF16	DSA (sign)	1024 bit	100 ms
SGS ST16CF54B	DSA (sign)	512 bit	163 ms
SGS ST16CF54B	RSA (sign)	512 bit	389 ms

B. Security of the Proposed Approach

Our previous analysis shows that to deceive the verifier, the attacker needs to compromise both the TRS and the smart card. In the following analysis, we will explain the potential threats when the attacker compromises only one of the two devices.

If the malicious attacker compromises a smart card, it can prevent a TRS from passing the attestation procedure. The malicious smart card can embed a very long attestation duration in its message to the verifier and the attestation will fail. This operation, however, will only prevent the attacker from using the TRS. When an attestation fails, the verifier cannot distinguish a compromised TRS from an infected smart card. Therefore, we propose to adopt a stateless approach: the verifier will not maintain a record of the previous attestation results. Whether or not a telesurgical system is diagnosed as 'healthy' is solely based on the attestation result of the current round. In this way, when a healthy smart card is inserted, the TRS will pass the attestation.

If the TRS has been compromised, it can try different schemes to impact the time measurement results of the smart card. For example, since the smart card can communicate to the verifier only through the TRS, the attacker can discard all data packets between the two entities. This action, however, will lead to the timeout of an attestation procedure. Similarly, the TRS can provide many fake message (3) in the protocol to the smart card and the smart card will be overwhelmed by the computation overhead. This denial-of-service attack will prevent the smart card from assisting the verifier in the attestation procedure. However, it cannot let a compromised TRS pass the integrity verification.

As a special attack on the smart card, the TRS can impact the behaviors of the smart card by adjusting its physical contact to the reader. For example, in [18] the research shows that the clock frequency of the smart card can be impacted when we adjust the interfaces between the card and the reader. When the frequency of the internal clock changes, the required time to generate or verify a digital signature will also change. Fortunately, when we examine the proposed attestation protocol, we find that these changes will not weaken its safety. Since we depend on the GPS signals instead of the internal clocks to measure the attestation duration of the TRS, reducing the power voltage or even cutting the power supply to the smart card will not buy the TRS more time to calculate the hash result. At the same time, the smart card measures the time duration between the seed and nonce are provided to the TRS and the hash result is returned back. During this procedure, the smart card does not need to conduct any computation. Therefore, the changed clock frequency will not impact the time measurement accuracy.

C. Return-oriented Programming Attack

A special attack that the software attestation mechanisms cannot defend against is the return-oriented programming attack [19], [20]. In this kind of attack, the malicious nodes do not inject code into the system. On the contrary, they use the stack of return addresses to chain together the code segments already in the system to form the malicious functions. For example, in [20], the attackers can use only the code segments in the legitimate programs to form a function to copy the malicious code back and force between the program memory and data memory. In this way, as soon as the attestation is accomplished, the malicious code will be copied back to the program memory. There have been mechanisms [21] designed to defend against such attacks. We plan to integrate our approach with those methods to provide a better protection to the TRS.

IV. RELATED WORK

A. Telesurgical Robot Security

While the majority of the research efforts on telesurgical systems focus on turning the technique into a practical method, some efforts have been conducted to improve the system security. In [3], the researchers integrate the public key cryptography mechanisms into TRS. In their approach, the controller and the telesurgical system use digital certificates based on X.509 to achieve authentication and use TLS/DTLS protocols to protect the TCP and UDP traffic. They propose the Interoperable Telesurgery Protocol (ITP), which is an open framework for establishing secure communications between the robot and the operator.

B. Remote System Attestation

The problem of remote system attestation has attracted a large amount of research efforts since the proliferation of embedded devices in pervasive computing. Both hardware and software-based approaches have been designed to achieve the goal. For example, the IBM cryptographic coprocessor [22] is designed to protect the software stack. The systems such as TCG [23] and NGSCB [24] use secure hash functions to verify the system integrity and store the results in a secure coprocessor. In [25], the Micro-Controller Units running at a higher privilege are used to protect the system integrity. Although the tamper-proof hardware will provide a solid foundation for the attestation mechanisms, it will introduce extra cost to the manufacture and maintenance of the robots. Therefore, a pure software based approach is preferred.

The software based approaches usually depend on the delay of the verification procedure caused by the malware to detect

an integrity violation. In SWATT [7], the attestation function randomly reads the program memory locations to force the malware to embed the "if" operations into hash calculation. The large number of "if" operations will introduce a noticeable delay into the attestation procedure. In [8], [9], the empty space in the device memory is filled with pseudo random numbers to prevent the malicious code from using the space. The code self-modification technique [26] has been used to mitigate the attack on the attestation function.

C. Smart Card in Security Applications

The smart cards have been widely used for authentication [27] and key distribution [28] in electronic commerce. To compromise the security of this specially designed hardware, the malicious users have tried various attacks on power analysis [29] and electromagnetic radiations [30]. In this paper, we use the smart cards as an independent device for time measurement. Since the smart card uses satellite based clocks and draws only power from the host device, it will be very difficult for the compromised TRS to manipulate the measurement results.

V. Conclusion

Telesurgical robot systems have important military usage in the future. In this paper we present a new mechanism that allows the verifier to remotely attest the integrity of the software system on the TRS. Using the smart card of the medical personnel as an independent device to measure the attestation duration, we can effectively separate it from the network transmission delay and diminish the impacts on the detection accuracy. We study the overhead of the proposed approach and investigate its security.

We are collaborating with the researchers at UT Dallas to extend our approach in the following aspects. First, we will integrate the software attestation mechanism into the ITP protocol to provide comprehensive protection of the TRS. Second, we will implement our approach in the simulator of the Raven Surgical Robot and test its performance on real networks. Finally, we plan to evaluate the smart card assisted attestation mechanism in several other network applications with very long transmission delays.

Acknowledgment

This research is supported in part by NSF award 0516085 and 0830624.

References

[1] B. Snethen and R. Steen, "Telesurgery pilot in Iraq enabled by MC4," US Army News Frontpage, 2009.

[2] N. Dowler and C. Hall, "Safety issues in telesurgery–summary," in *Proceedings of IEE Colloquium on Towards Telesurgery*, 1995.

[3] G. Lee and B. Thuraisingham, "Cyber physical systems security applied to surgical robotics," UT Dallas, submitted under review at Computer Standards and Interfaces, Tech. Rep., 2010.

[4] D. S. Stefanos, S. Gritzalis, J. Iliadis, D. Gritzalis, and S. Katsikas, "Trusted third party services for deploying secure telemedical applications over the www," in *the WWW, Computers & Security*, 1999, pp. 627–639.

[5] F. Wozak, T. Schabetsberger, and E. Ammmenwerth, "End-to-end security in telemedical networks–a practical guideline," *International Journal of Medical Informatics*, vol. 76, no. 5, pp. 484–490, 2007.

[6] Y. Yang, F. Bao, and R. Deng, "Secure an image-based simulated telesurgery system," in *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, 2003.

[7] A. Seshadri, A. Perrig, L. V. Doorn, and P. Khosla, "Swatt: Software-based attestation for embedded devices," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2004.

[8] Y.-G. Choi, J. Kang, and D. Nyang, "Proactive code verification protocol in wireless sensor network," in *ICCSA*, 2007.

[9] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proceedings of IEEE International Symposium on Reliable Distributed Systems*, 2007, pp. 219–230.

[10] M. Lum, J. Rosen, H. King, D. Friedman, and et al, "Telesurgery via unmanned aerial vehicle (uav) with a field deployable surgical robot," in *Proceedings of Medicine Meets Virtual Reality*, 2007, pp. 313–315.

[11] R. Jenkins, "Isaac," in *International Workshop on Fast Software Encryption*, 1996, pp. 41–49.

[12] Y. Lin, X. Maozhi, and Z. Zhiming, "Digital signature systems based on smart card and fingerprint feature," *Journal of Systems Engineering and Electronics*, vol. 18, no. 4, pp. 825–834, 2007.

[13] "A-gps - the world's first assited gps sim card," Sagem Orga Group press news, 2007.

[14] M. Lum, D. Friedman, J. Rosen, G. Sankaranarayanan, H. King, K. Fodero, R. Leuschke, M. Sinanan, and B. Hannaford, "The RAVEN - design and validation of a telesurgery system," *International Journal of Robotics Research*, vol. 28, pp. 1183–1197, 2009.

[15] B. Blanchet, "Automatic verification of correspondences for security protocols," *J. Comput. Secur.*, vol. 17, no. 4, pp. 363–434, 2009.

[16] A. van Wynsberghe and C. Gastmans, "Telesurgery: an ethical appraisal," *J. Med Ethics*, vol. 34, no. 10, 2008.

[17] H. Handschuh and P. Paillier, "Smart card crypto-coprocessors for public-key cryptography," in *Smart Card Research and Applications*, J.-J. Quisquater and B. Schneier, Eds., 2000, pp. 386–394.

[18] H. Bar-El, "Known attacks against smartcards," Discretix Technologies Ltd. Tech Report, 2005.

[19] R. Hund, T. Holz, and F. Freiling, "Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms," in *Proceedings of the USENIX Security Symposium*, 2009.

[20] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, "On the difficulty of software-based attestation of embedded devices," in *Proceedings of ACM conference on Computer and communications security*, 2009, pp. 400–409.

[21] L. Davi, A.-R. Sadeghi, and M. Winandy, "Dynamic integrity measurement and attestation: towards defense against return-oriented programming attacks," in *Proceedings of the ACM workshop on Scalable trusted computing*, 2009, pp. 49–54.

[22] S. W. Smith and S. Weingart, "Building a high-performance, programmable secure coprocessor," *Comput. Netw.*, vol. 31, no. 9, pp. 831–860, 1999.

[23] "Trusted computing group (tcg)," https://www.trustedcomputinggroup.org/, 2003.

[24] "Next-generation secure computing base (ngscb)," http://www.microsoft.com/resources/ngscb/ default.mspx, 2003.

[25] M. LeMay and C. A. Gunter, "Cumulative attestation kernels for embedded systems." in *European Symposium on Research in Computer Security*, 2009, pp. 655–670.

[26] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote software-based attestation for wireless sensors," in *ESAS*, 2005.

[27] B. Preneel, "A survey of recent developments in cryptographic algorithms for smart cards," *Computer Networks*, vol. 51, no. 9, pp. 2223 – 2233, 2007.

[28] W.-S. Juang, "Efficient multi-server password authenticated key agreement using smart cards," *IEEE TRANSACTIONS ON CONSUMER ELECTRONICS*, vol. 50, pp. 251–255, 2004.

[29] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks - Revealing the Secrets of Smartcards*. Springer, 2007.

[30] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*, 2001, pp. 200–210.