THE USE OF CRYPTOGRAPHIC PRINCIPLES WITHIN METROLOGY SOFTWARE

SIMON PARKINSON, ANDREW P. LONGSTAFF, ANDREW CRAMPTON, GARY ALLEN, SIMON FLETCHER AND ALAN MYERS

Centre for Precision Technologies, University of Huddersfield, Queensgate, Huddersfield, West Yorkshire, HD1 3DH, UK

Typically the design and production of metrology software is based upon a rigorous process of establishing the software requirements. Both the metrology and security requirements will be processed and normally evaluated in isolation, or at best simultaneously. With the use of cryptographic principles, and the more prominent object-oriented programming languages, a new security-centric philosophy to metrology software design and production is presented to provide secure, robust and functional metrology software.

1. Introduction

To enable a thorough software design, a rigorous process of establishing the desired requirements must be performed. This methodology is certainly used for the production of metrology software. When establishing the requirements, both metrology and security aspects will be considered to allow for design and production of robust, secure and functional software. This will typically result in effort being spent to satisfy both metrology and security requirements separately. An illustration of the current effort is shown in Figure 1A

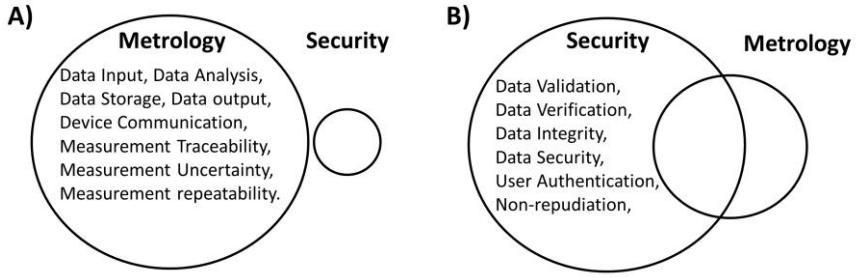

Figure 1. Showing the design effort of the current and proposed methodology. The effort is represented by the size of the circle.

In this paper we present a fundamental shift in philosophy to the design and production of metrology software, which is illustrated in Figure 1B. This new methodology involves incorporating the fundamental cryptographic principles

within the design and development of metrology software to provide a proven method of security, and a viable method of complying with the metrology-related functional requirements, which are explained in Section 2.

2. Requirements

2.1. *2.1 Basic metrology*

There is a common set of metrology related functional requirements [1], with differences regarding the connectivity and distribution of the software. The following list summarizes the common metrology specific functional requirements.

- Traceability – All aspects of the measurement should be traceable, and the software should record all the necessary information where possible to ensure this. A good example of this can be seen in the Renishaw Ballbar 20 software [2] where the serial number, time of last calibration and the calibration certificate number of the connected instrumentations are logged.
- Uncertainty – All uncertainties must be correctly estimated and recorded to ensure that the measured value is meaningful. Great effort has been spent by many to create supporting software [3] to the Guide to the Expression of Uncertainty in Measurement (GUM) and overcoming the complexities of validating GUM conformity [4].
- Repeatability –The software should promote repeatability by recording all the necessary information to allow for the measurement to be performed in exactly the same conditions.

2.2. *2.2 Basic security*

The security requirements of an internet-enabled metrology system are well explored [5]. However, the security requirements for software that makes use of a different networking medium, or operates in a standalone manner, should be given the same emphasis to ensure that adequate security precautions are always taken. The following list summarizes the main software security requirements.

- Data security – Security is paramount, especially with commercially sensitive data which might be transferred publically. This need is well established throughout the information technology community. However, so is the complexity associated with performing a data security risk analysis.
- Data integrity – Correctly identifying accidental and malicious data modification is essential for establishing a high level of data confidence.
- Data authentication – Establishing the authenticity of the data and sender is critical for maintaining traceability and repeatability.

- User authentication – Stopping unwanted access to the metrology system is necessary to protect sensitive data and equipment.

3. Security-centric design philosophy

The new philosophy proposed in this paper is a security-centric approach to the design and production of metrology software. As illustrated in Figure 2, by carrying out thorough design and implementation of the security requirement, the functionality requirements can also be satisfied. This strong relationship between the security and metrology requirements is underpinned by the captured metrology data. For example, information making the performed measurement repeatable and traceable, which includes all processing comparisons and uncertainties, should be recorded within the data. This means that employing good security procedures will maintain the integrity of the metrology data, thus preserving its qualities.

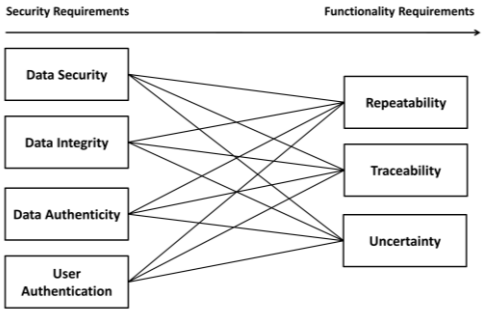

Figure 2. Illustration showing the relationship between the security and the metrology requirements of a software-based metrology system

4. Cryptographic Principles

Previous efforts have incorporated the use of cryptographic functionality in a bottom up approach from design to implementation [5, 6]. Here we propose a similar procedure, however, now we are concentrating on satisfying the security requirements first in the knowledge that by doing so we are also satisfying the metrology requirements. The asymmetric public-key infrastructure provides a function set that can satisfy all the security requirements of metrology software. The following list states the cryptographic functionality that can be implemented to meet the security requirements.

- Data security – Using the asymmetric infrastructure it is possible to encrypt information using a recipient's public-key which can subsequently only be decrypted with the matching private-key [7]. There are many different

encryption algorithms that have slightly different computational requirements [8]. However, with the decreasing cost of computations power, any performance disadvantages are significantly reduced.

- Data integrity – With the use of a one way hash function, a unique hash value can be produced for a given data set and can be used for integrity checking at any time. By incorporating, for example, the measurement timestamp traceability to a particular measurement can be guaranteed.
- Data Authenticity - The same public-key infrastructure allows for information to be signed by an individual's private-key, which can subsequently be verified by anyone with access to the individual's matching public-key.
- User Authentication – Digital signatures can be used as an alternative method of authentication from specifying a username and password. An example of this would be where a user creates a digital signature using their private key, which can then be programmatically verified by using the user's public key.

5. Implementation

Implementing cryptographic functionality within many programming languages has been made possible by pre-written and tested programming libraries within the languages Application Programming Interface (API). The programming language of Java contains the Java Cryptography Architecture, and similarly, the Microsoft .NET framework provides the System.Security.Cryptography namespace.

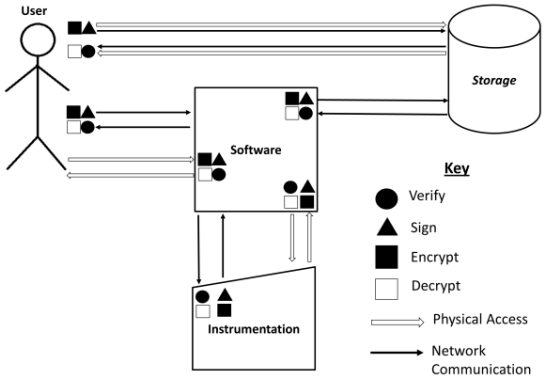

Figure 3. Illustration of where to insert the cryptographic functionality within a software system

The implementation of cryptographic functionality within well-established Object-Oriented (OO) languages like Java and those in the .NET framework is

not programmatically difficult and can be implemented with little additional cost. If programmed using an OO language, the cryptographic functions can be used globally throughout the software as frequently as required. Figure 3 shows the many locations where a software engineer might consider implementing cryptographic techniques within the metrology software.

6. Conclusion

In this paper we have demonstrated the relationship between the security and metrology software requirements. We have presented the philosophy of a security-centric approach to the design and implementation of metrology software. Following this philosophy, and the use of cryptographic functionality within an object-oriented language, can save on programming effort to achieve secure, reliable and functional software. However, further work will be performed to validate the feasibility of the presented philosophy for both small and large scale software development projects.

7. References

1 BS, Measurement management systems - Requirements for measurement processes and measuring equipment. 2003, BSI: London, United Kingdom.

2. Renishaw. *System software for QC20-W ballbar*. 2011 [cited 2011; Available from: http://www.renishaw.com/en/system-software-for-qc20-w-ballbar--11076

3. G. Norbert, S. Heike, R. Dieter, Software validation in metrology: A case study for a GUM-supporting software. Measurement, 2006. **39**(9): p. 849-855.

4. B.Wichmann, G. Parkin, R. Barker, Software Support for Metrology Best Practice Guide No. 1, in Validation of Software in Measurement Systems. 2007, National Physical Laboratory.

5. R. M. Barker., Software Support for Metrology Best Practice Guide No. 19, in Internet-enabled Metrology Systems. 2006, National Physical Laboratory

6. Å. Sand, H. Slinde, T. A. Fjeldly, A Secure Approach to Distributed Internet-Enabled Metrology. IEEE Transactions on Instrumentation and Measurement, 2007. **56**(5): p. 1979 - 1985

7. IEEE, Standard Specifications for Public-Key Cryptography. 2000, IEEE.

8. D. S. A. Elminaam, H. M. A. Kader, M. M. Hadhoud, Evaluating The Performance of Symmetric Encryption Algorithms. International Journal of Network Security, 2010. **10**(3): p. 213-219.