Using Bloom Filters to Ensure Access Control and Authentication Requirements for SCADA Field Devices

Jeffrey Hieb, Jacob Schreiver, James Graham

Chapter 7

USING BLOOM FILTERS TO ENSURE ACCESS CONTROL AND AUTHENTICATION REQUIREMENTS FOR SCADA FIELD DEVICES

Jeffrey Hieb, Jacob Schreiver and James Graham

Abstract The critical infrastructure cannot operate without SCADA systems; this has made the task of securing SCADA systems a national security priority. While progress has been made in securing control networks, security at the field device level is still lacking. Field devices present unique security challenges and these challenges are compounded by the presence of legacy devices. This paper describes a technique that uses Bloom filters to implement challenge-response authentication and role-based access control in field devices. The approach, which is implemented in an inline security pre-processor, provides for rapid and constant access check times. Experiments involving a prototype device demonstrate that the false positive rate can be kept arbitrarily low and that the real-time performance is acceptable for many SCADA applications.

Keywords: SCADA systems, field devices, security, Bloom filter

1. Introduction

Supervisory control and data acquisition (SCADA) systems and distributed control systems (DCS) are networks of computer systems that provide remote telemetry and control of physical systems and processes. Collectively they are referred to as industrial control systems (ICSs). ICSs play a central role in the operation of many critical infrastructures such as electric power, drinking water, waste water treatment, oil and gas distribution, and industrial manufacturing. A typical ICS comprises a master, one or more field devices and a communications infrastructure. The master or master terminal unit (MTU) processes information received from field devices, presents the information to operators and engineers via a human machine interface (HMI), and sends control directives to field devices. The master and field devices are connected via

a communications network, which may include leased serial lines, telephone circuits, cellular networks and UHF/VHF radio. The communication protocols used by the master and field devices are referred to as SCADA protocols.

When these systems were originally deployed, little attention was paid to securing them because the systems were physically isolated and used proprietary hardware, software and communication protocols [2–4, 7, 9]. However, due to network connectivity and convergence, these systems have become highly vulnerable to cyber attacks [7, 9]. Many of the initial efforts to secure these systems use traditional approaches such as firewalls and network intrusion detection systems. While this has been an appropriate first response, Stuxnet and other recent threats underscore the importance of also securing field devices.

We have worked on the task of securing field devices for several years. Our initial effort involved the development of a security-hardened architecture for field devices. Our recent work has focused on developing an in-line security solution for legacy devices using a microkernel based security-hardened architecture. This device, which we call a field device security pre-processor (FD-SPP), provides authentication and role-based access control (RBAC) for legacy field devices. Enforcing RBAC requires checking whether or not a message or operation is allowed, but this involves multiple set membership checks that can have a negative impact on performance.

This paper presents a novel approach using Bloom filters that speeds up field-device-level access control checks to prevent interference with process control operations. In particular, a dual Bloom filter structure is used to minimize security processing while reducing and quantifying the risk of potential attacks. The resulting FD-SPP provides two key security features: authentication using challenge-response authentication and role-based access control enforcement.

2. Background

The field device security pre-processor (FD-SPP) is an in-line device for securing field devices. The FD-SPP uses a security-hardened field device architecture proposed by Hieb, *et al.* [6]. A key advantage of this architecture is that it supports formal verification techniques. The FD-SPP is placed in front of a legacy field device by connecting the communication network interface to the FD-SPP and then connecting the FD-SPP to the field device. A software component running on the MTU/HMI or an external hardware device similar to the FD-SPP works with the FD-SPP to implement security functionality. Figure 1 shows the placement of a FD-SPP in a simple SCADA environment.

To be effective, the FD-SPP needs to implement its security features so that performance is maximized. In addition, the implementation needs to support formal verification techniques. Bloom filters provide a means to achieve both goals. A brief description of the challenge-response authentication scheme is described in Section 2.1 and an overview of the role-based access control technique is presented in Section 2.2. To provide maximum performance, Bloom filters are used to determine if a message is to be challenged and if a received

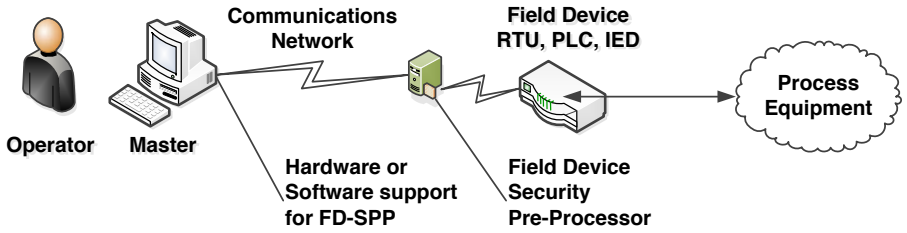

Figure 1. Placement of the FD-SPP in a simple industrial control system.

operation is allowed for the associated user. Section 2.3 provides a brief introduction to Bloom filters.

2.1 FD-SPP Challenge Response Authentication

The incorporation of challenge-response authentication in SCADA protocols is described in detail by Patel [11]. In challenge-response authentication, communicating parties, in this case, an operator or engineer using an MTU/HMI or engineering workstation, and the FD-SPP share a secret (one shared secret for each operator or engineer). When the FD-SPP receives a message from the MTU or workstation, it issues a challenge message in response. The challenge message incorporates a nonce to prevent replay attacks. The MTU/HMI or workstation then builds a challenge-response message, which includes the previous message sent, user identity information, and a hash-based message authentication code (HMAC) that is computed for the message. The HMAC is generated using SHA-256. The HMAC incorporates the shared secret, so only the operator/engineer using the MTU/HMI or workstation can correctly generate the HMAC. The FD-SPP checks the HMAC against the HMAC it calculates. If the two HMACs match, then the message is authenticated and forwarded to the field device.

Due to the nature of ICSs and SCADA protocols, not every message needs to be challenged. For example, reading a coil or analog input has no effect on the field device state, so it is reasonable to have a policy that does not require all messages to be challenged. At runtime, the task of determining whether or not a received SCADA message must be challenged is the responsibility of the access control system discussed in the next section.

2.2 FD-SPP Access Control

In addition to authentication, the FD-SPP provides a simple role-based access control system. In role-based access control (RBAC), users are assigned roles and privileges or capabilities are assigned to the roles. A user may only perform operations assigned to the role possessed by the user. In an ICS setting, there may be different roles for operators, engineers, security administrators and vendors. Grouping privileges or operations by role makes it easier to manage them. For the FD-SPP, each user is assigned a single role, e.g., "op-

erator." When the access control system receives a SCADA message, it must make two decisions: (i) whether or not the message needs to be challenged; and (ii) whether or not the requested operation is to be allowed. Making this decision involves consulting the FD-SPP security policy, which must be developed for each installation by operators, engineers and security administrators. The policy must define roles, assign allowed operations to roles and assign users to roles. If a user attempts to perform an operation that is not associated with the role to which the user is assigned, then the access control system should deny the operation.

2.3 Bloom Filters

A Bloom filter is a probabilistic data structure for determining set membership [1]. Bloom filters have space and time advantages that render them an attractive approach for controlling access to SCADA field devices. The space advantage comes from the fact that a Bloom filter maintains its size no matter how many elements are added to the set. The time advantage arises because there are no loop structures that depend on n (number of elements in the set) to determine if a given element is a member of the set. However, the space and time advantages yield a major disadvantage, false positive errors.

A Bloom filter begins as an empty array of m bits. This empty Bloom filter returns false when any element is checked for membership in the filter. To add an element to the Bloom filter, an element is first passed through k hash functions. The result of each of these hash functions is used to create a position in the Bloom filter array; the bit at each of these positions is set to one. In order to check if an element is in the Bloom filter, the same hash functions are used to create k positions in the Bloom filter. If all the positions in the array have a one, the object is said to be in the Bloom filter. Because of the use of hash functions to add and check entries in the array, collisions in the hash function outputs cause false positive errors. For large values of m, the false positive rate p of a Bloom filter is given by:

$$p = \left(1 - e^{-kn/m}\right)^k. \tag{1}$$

Given n elements in a Bloom filter, the values of k and m can be chosen to achieve any desired false positive rate. In many implementations, m is the nearest power of 2 and k is rounded to the nearest integer m [12].

3. Using a Dual Bloom Filter

In the FD-SPP, Bloom filters are used in two ways: (i) to determine if a requested operation is to be challenged; and (ii) if a requested operation is allowed for the user (assigned to a role) making the request. Initial requests from a source are always challenged by the FD-SPP. After the initial challenge has been met, subsequent operations that are requested are checked to determine if: (i) the user is allowed to request the operation; and (ii) if the operation is

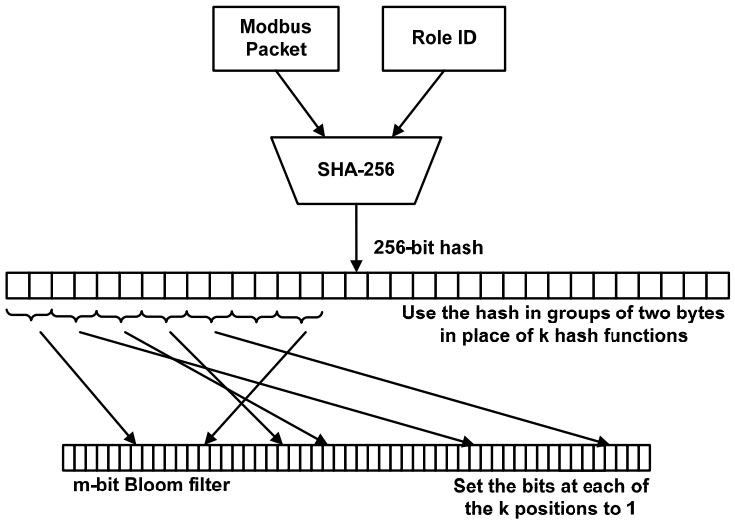

Figure 2. Inserting a Modbus packet in a Bloom filter.

critical and needs to be challenged. The need for quick and efficient processing of SCADA messages by the FD-SPP leads to the possibility of using a Bloom filter to implement the access control checks. The popular Modbus protocol [10] is used for development and testing purposes.

The first step is to create a Bloom filter that implements a given RBAC policy for Modbus operations. We begin by selecting a target false positive rate; in our case, a target false positive rate of 0.01 was used. Next, the bit field size (m) is chosen given the number of entries to be added to the filter. For the given policy, there is an entry for every <role, Modbus Operation> tuple explicit in the policy. For example, if $n = 100$, then $m = 1024$ (nearest power of 2 to 958.5) and $k = 7 \cong 7.0979$.

To add an element to the Bloom filter, the packet and the role ID are combined and hashed using SHA-256. The resulting hash is broken up in order to serve as seven hash functions for the Bloom filter (k=7). The first two bytes of the hash serve as the first hash function, the second two bytes serve as the second hash function, and so on until the seventh hash. Since two bytes are more than necessary to generate a number from 0 to 1023, the lower ten bits of each set of two bytes are used to create the position in the Bloom filter. This scheme has been used by Tripunitara and Carbunar [13]. Using this scheme, role Modbus messages in the policy are added to the Bloom filter one by one as shown in Figure 2.

When a message is received by the FD-SPP, it looks up the user role and hashes the entire message and role to check if the operation is allowed. The sizes and number of hash functions can be varied based on the number of packets and roles desired for the RBAC policy.

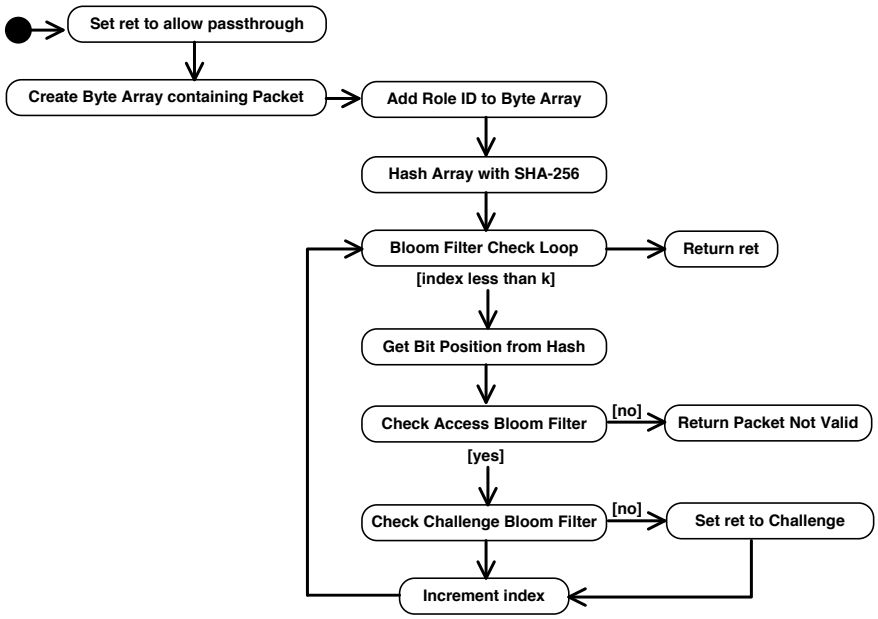

Figure 3. Flow diagram of the access control logic.

A second Bloom filter is used to determine whether or not a message must be challenged. The second Bloom filter can be implemented in two different ways: as a filter containing packets to challenge or as filter containing packets not to challenge. Our rationale for choosing the second approach is provided in Section 4. The second filter has the same number of bits and uses the same hashes as the first filter. However, entries are added to the second filter only if they are not to be challenged. When a requested operation is checked against this Bloom filter, it is hashed in the same way as when adding a new element. The positions are checked in the RBAC Bloom filter first. If all the bits are not equal to one, then the packet is rejected by the RBAC filter. This filter contains all the allowed operations of the field device, which is why it is checked first. If the packet is not rejected, then the same positions are checked in the second Bloom filter. If all the bits in this filter are equal to one, then the packet is allowed to pass on to the field device; if not, a challenge is issued, which authenticates the user requesting the operation. Pass through is allowed in cases where the message is not critical and the sender of the message has recently responded to a challenge. The diagram in Figure 3 presents the procedure for checking entries in the Bloom filter.

The advantage of this approach is that constant time is required for the FD-SPP to check if a given role is allowed to carry out a specific operation and if a message is a critical operation. This is important because it leads to improved processing time for the FD-SPP. However, as described above, a

disadvantage is the false positive rate of Bloom filters. The evaluation in the following section considers performance and the false positive rate.

4. Evaluation

The evaluation was conducted using an FD-SPP prototype for which the Bloom filter approach was implemented on a Gumstix Verdex Pro PXA 270 XScale processor with 64 MB RAM. A simple HMI/MTU was implemented on a separate personal computer using LabView. A virtual serial device was used to place a software component between the MTU/HMI and the serial network interface to provide the complementary FD-SPP support on the MTU/HMI for challenge-response authentication. Modbus messages were collected by sniffing Modbus traffic between the test MTU/HMI and an actual field device before security was added.

Two roles, operator and engineer, were used in the evaluation. Combining the collected Modbus operations with roles resulted in eighteen <role, Modbus Operation> pairs that had to be added to the Bloom filter. Two of the <role, Modbus Operation> message pairs were considered non-critical and were added to the second Bloom filter to indicate that they did not have to be challenged.

4.1 Performance

Bloom filter access checks take a very small amount of time, even when running on minimal hardware. In the case of the Gumstix Verdex Pro XM4 COM, the time required to perform the hash calculation for an element and compare the value with the Bloom filter was about $18\mu s$. On the same device, it took about $15\mu s$ to perform a SHA-256 hash; this time includes only the internal Bloom filter look-up time, not the round trip time from the HMI to the field device. Real-world SCADA installations may require more elements to be added to the Bloom filter, since their policies are likely to be more complicated. However, due to the time-invariant scalability of the Bloom filter, it should be possible to increase the size of the Bloom filter by several orders of magnitude without effecting the access check time. The round trip time was approximately 300ms, which is acceptable in many SCADA applications.

4.2 False Positives

A key issue to be considered when using Bloom filters is the presence of false positives. In this application, a false positive could correspond to a situation where a forged Modbus message inserted by an attacker is accepted unchallenged by the FD-SPP. The false positive rate of the Bloom filter structure indicates the difficulty (or likelihood) that such a message could be found by an attacker. A false positive rate of zero is ideal, but this is not possible with Bloom filters. Instead, the false positive rate must be kept as low as possible while maintaining the speed at which an access check can be performed.

Recall that there are two possibilities in the case of the second Bloom filter: challenged packets are in the Bloom filter or non-challenged packets are in the filter. When the packets are not to be challenged, the false positive rate of the system for non-challenged false positives (successful attacks) is equal to the false positive rate of the second filter. To define this rate, let c represent the number of packets to be challenged, n be the number of packets added to the system, m the length in bits of the filter, k the number of hash functions used, p the false positive rate used to create the first Bloom filter, and r the ratio of packets to be challenged over the total number of packets (c/n). Then, the false positive rate for a Bloom filter containing non-challenged packets is given by:

$$\begin{aligned} p_{non-challenge} &= \left(1 - e^{(\frac{-k(n-c)}{m})}\right)^k = \left(1 - 2^{(\frac{c-n}{n})}\right)^{\frac{ln(2)m}{n}} \\ &= \left(1 - 2^{r-1}\right)^{-\frac{ln(p)}{ln(2)}} \end{aligned}$$

Alternatively, for the Bloom filter containing entries to be challenged, a non-challenged false positive can occur when the packet is accepted by the first filter but rejected by the second. This means that it must have at least one bit that is in the first Bloom filter but not in the second. The maximum odds of this occurring can be calculated as the probability of all but one bit being any of the ones in the first filter multiplied by the ratio of the difference of the number of ones between the filters over the length in bits of the filters. This is given by:

$$\begin{aligned} p_{challenge} &= (1 - e^{\frac{-kn}{m}})^{k-1} * \frac{(m(1 - e^{\frac{-kn}{m}}) - m(1 - e^{\frac{-kc}{m}}))}{m} \\ &= \left(\frac{1}{2}\right)^{\frac{((mln(2))}{(n-1)}} * \left(-\frac{1}{2} + 2^{(\frac{-c}{n})}\right) \\ &= p(-1 + 2^{(\frac{n-c}{n})}) = p(-1 + 2^{1-r}). \end{aligned}$$

This equation shows that the Bloom filter with non-challenged entries has a lower false positive rate because it scales exponentially with the false positive rate of the first Bloom filter while the challenge Bloom filter scales linearly.

After adding 18 entries to the first Bloom filter and two entries to the second Bloom filter (non-challenged operations), it is possible to accurately determine the false positive rate for the specific Bloom filter structure used in this evaluation. Using $n = 18$ in Equation (1), the estimate for the false positive rate is $8.5172 * 10^{-14}$. This is merely the theoretical false positive rate of the approximation of the Bloom filter after 18 entries. The number of ones in the Bloom filter can be used to calculate the actual false positive rate of the Bloom filter. The access control Bloom filter, with $m = 1024$ bits, has 14 bits set to one. Therefore, the probability of any single bit being one is simply 14/1024,

making the actual probability of a false positive in the access control Bloom filter equal to:

$$(\frac{14}{1024})^{7} = 8.9289 * 10^{-14}.$$

The existence of false positives, no matter how small the rate, may appear to indicate that the approach is inappropriate for the FD-SPP access control check. However, we argue that a sufficiently low false positive rate can provide a level of security similar to other techniques. For example, consider the use of symmetric encryption with a key length of n. Brute force attempts to find the key work for sufficiently small n. Similarly, the use of a Bloom filter has a false positive rate relative to k, m and n.

Fortunately, there are several approaches for reducing the false positive rate of the Bloom filter used in this application. The approaches include increasing the size of the Bloom filter and changing the number of hash functions. We assume that the numbers of roles and messages are known before the system is implemented; therefore, the number of bits in the Bloom filter and the number of hash functions can be selected to achieve any desired false positive rate greater than zero. There is also a way to reduce the false positive rate without changing the number of bits or the number of hash functions; this approach is discussed in Section 4.3.

While the false positive rate is central to a security analysis of our approach, there is an important difference between the false positive rate of the Bloom filter structure and the effort required to brute-force a cryptographic secret. If an attacker finds a Modbus message that makes it through the Bloom filter, only this message can be used in an attack (a successful attack would most likely require multiple messages). Additionally, there are only so many Modbus messages that could actually damage a system. For example, only a limited number of messages can write to a particular critical coil. Since the Bloom filter can be checked in advance, analysis can be performed to verify that no messages that can damage the system are non-challenged false positives.

Also, since the attacker would not have access to the Bloom filter, he/she would have to attack the system directly. In the case of the prototype, it takes approximately 200ms to receive a challenge from the security pre-processor. This means that the attacker would have to wait at least 200ms between attempts. If the attacker were to attack the system for an entire day, he/she would be able to perform 432,000 attacks. In Section 4.3, we will show that the actual false positive rate for the implementation can be reduced to $1.65 * 10^{-14}$. Using this final false positive rate, the probability of a successful brute force attack that identifies a single message that could bypass the Bloom filter after one day is just $7.13 * 10^{-9}$.

4.3 Reducing the False Positive Rate

In order to reduce the false positive rate, it is important to first identify the variables that are related to the false positive rate of the Bloom filter. Simply put, if two Bloom filters have the same number of hash functions k and the

same number of bits m, then the only thing that can make the false positive rate any different is the number of ones in the filter. The number of entries in a Bloom filter is based on the number of elements added to the filter multiplied by the number of hash functions minus the number of collisions. Therefore, increasing the number of collisions results in a Bloom filter with a lower false positive rate. Note that this does not imply the use of hash functions that create more collisions; instead, it means using hash functions that collide for the specific values that are added to the Bloom filter. The hash functions must still have uniform results for arbitrary input data, otherwise the bias would yield more false positives, not less. This approach was first described by Hao and colleagues [5].

For example, suppose that two entries A and B are placed in a Bloom filter that uses seven hash functions. Each entry adds seven bits to the Bloom filter, for a total of 14 bits set. Assume that a list of hash functions exists from which the seven hash functions with the most collisions can be selected. In the case of a single collision, 13 bits are added to the Bloom filter instead of 14. Since this value is raised to the power k, the number of collisions can have a large effect on the false positive rate:

$$\frac{13^7}{14^7} = 0.59526.$$

In this case, adding a single collision reduces the false positive rate to nearly 60% of its previous value.

In the more general case, let x be the number of entries in the filter and c the collision percentage that can be invoked. Then,

$$(\frac{x}{m})^k$$

can be reduced to:

$$(\frac{x(1-c)}{m})^k.$$

This means that the false positive rate can be reduced by:

$$1-(1-c)^k.$$

For example, a collision rate of 10% for the known entries of the Bloom filter reduces the false positive rate by more than 50%.

The prototype evaluation system described at the beginning of this section uses seven hash functions, the Bloom filter has 1,024 bits and 14 bits were set to one when the 18 elements were added. This yields the following false positive rate for non-challenged entries:

$$p = (\frac{14}{1024})^7 = 8.9289 * 10^{-14}.$$

Upon searching for collisions between two entries in the second Bloom filter, seven new hash functions with uniform distributions were found. These hash functions had three collisions for the two entries in the challenge-response Bloom filter. This reduces the number of ones from 14 to 11. This yields a collision rate of 0.81514 and more than 80% reduction in the false positive rate using the same number of hash functions and entries, and the same bit length. The new false positive rate for non-challenged false positives is given by:

$$p = (\frac{11}{1024})^7 = 1.65 * 10^{-14}.$$

The new hash functions used are based on the SHA-256 algorithm as in the previous case, so no additional cost for using these hash partitions is imposed.

Since the original Bloom filter was designed for 100 packets whereas only 18 were actually used, the Bloom filter can be further optimized in theory. Using the function above for calculating the number of hash functions shows that 39 hash functions is the optimal number. This number of hash functions produces a false positive rate of $2.4547 * 10^{-44}$. Assuming the worst case situation of no internal collisions, this Bloom filter provides stronger security than a 144-bit cryptographic secret.

Table 1 presents the m and k values that can be used to achieve the targeted false positive rates for different numbers of system messages that would need to be added to an access control Bloom filter structure as described in Section 3. The table entries demonstrate that the approach is viable for SCADA systems with larger numbers of message-role pairs and that acceptable false positive rates can be achieved for these systems.

5. Conclusions

Securing legacy field devices is a challenging task because they have long deployment lifetimes and lack the processing power and memory required to implement security solutions. The field device security pre-processor (FD-SPP), which provides authentication and role-based access control for legacy devices, is a promising in-line security solution for legacy field devices.

The dual Bloom filter approach presented in this paper speeds up access control checks by the FD-SPP to prevent interference with process control operations. In particular, the structure is able to make two access control decisions: (i) whether a message (requested by a user) is allowed or denied; and (ii) whether or not the message is critical and should be challenged. Another advantage of the Bloom filter implementation is that it facilitates the verification of the operation of the entire device. However, a key drawback with the use of a Bloom filter is that it introduces false positive errors. While the errors cannot be eliminated, the analysis indicates that the false positive rate can be made arbitrarily low and that, for sufficiently low false positive rates, the approach is as secure as an n-bit key shared between two parties.

Our future research will focus on the formal verification of access checks and the security code used by the FD-SPP. If formal verification is possible,

Table 1. Parameter values for achieving various false positive rates.

Desired FP Rate	Messages	Percent Challenged	m	k	p
1.00E-13	100	0.50	3,516	24	1.17E-13
1.00E-13	200	0.50	7,033	24	1.16E-13
1.00E-13	300	0.50	10,550	24	1.16E-13
1.00E-13	400	0.50	14,067	24	1.16E-13
1.00E-13	500	0.50	17,584	24	1.16E-13
1.00E-13	100	0.75	2,349	16	1.30E-13
1.00E-13	200	0.75	4,698	16	1.30E-13
1.00E-13	300	0.75	7,047	16	1.30E-13
1.00E-13	400	0.75	9,397	16	1.30E-13
1.00E-13	500	0.75	11,746	16	1.30E-13
1.00E-13	100	0.90	1,597	11	1.14E-13
1.00E-13	200	0.90	3,194	11	1.14E-13
1.00E-13	300	0.90	4,792	11	1.13E-13
1.00E-13	400	0.90	6,389	11	1.13E-13
1.00E-13	500	0.90	7,986	11	1.13E-13
1.00E-20	100	0.50	5,410	37	1.22E-20
1.00E-20	200	0.50	10,821	37	1.22E-20
1.00E-20	300	0.50	16,231	37	1.22E-20
1.00E-20	400	0.50	21,642	37	1.22E-20
1.00E-20	500	0.50	27,052	37	1.22E-20
1.00E-20	100	0.75	3,614	25	1.05E-20
1.00E-20	200	0.75	7,228	25	1.05E-20
1.00E-20	300	0.75	10,842	25	1.05E-20
1.00E-20	400	0.75	14,457	25	1.05E-20
1.00E-20	500	0.75	18,071	25	1.05E-20
1.00E-20	100	0.90	2,457	17	1.06E-20
1.00E-20	200	0.90	4,914	17	1.06E-20
1.00E-20	300	0.90	7,372	17	1.06E-20
1.00E-20	400	0.90	9,829	17	1.06E-20
1.00E-20	500	0.90	12,287	17	1.06E-20

then the resulting quantified false positive rates for FD-SPPs could provide valuable input to risk assessment and risk management efforts for industrial control systems and the critical infrastructure assets in which they are used.

References

[1] B. Bloom, Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM*, vol. 13(7), pp. 422–426, 1970.

[2] T. Brown, Security in SCADA systems: How to handle the growing menace to process automation, *Computing and Control Engineering Journal*, vol. 16(3), pp. 42–47, 2005.

[3] M. Brundle and M. Naedele, Security for process control systems: An overview, *IEEE Security and Privacy*, vol. 6(6), pp. 24–29, 2008.

[4] D. Geer, Security of critical control systems sparks concern, *IEEE Computer*, vol. 39(1), pp. 20–23, 2006.

[5] F. Hao, M. Kodialam and T. Lakshman, Building high accuracy Bloom filters using partitioned hashing, *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 277–288, 2007.

[6] J. Hieb, S. Patel and J. Graham, Security enhancements for distributed control systems, in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi (Eds.), Boston, Massachusetts, pp. 133–146, 2007.

[7] V. Igure, S. Laughter and R. Williams, Security issues in SCADA networks, *Computers and Security*, vol. 25(7), pp. 498–506, 2006.

[8] R. Langner, Stuxnet: Dissecting a cyberwarfare weapon, *IEEE Security and Privacy*, vol. 9(3), pp. 49–51, 2011.

[9] A. Miller, Trends in process control systems security, *IEEE Security and Privacy*, vol. 3(5), pp. 57–60, 2005.

[10] Modbus Organization, Modbus Specification and Implementation Guides, Hopkinton, Massachusetts (`www.modbus.org/specs.php`), 2012.

[11] S. Patel, Secure Internet-Based Communication Protocol for SCADA Networks, Ph.D. Dissertation, Department of Computer Science and Engineering, University of Louisville, Louisville, Kentucky, 2006.

[12] Y. Qiao, T. Li and S. Chen, One memory access Bloom filters and their generalization, *Proceedings of the IEEE International Conference on Computer Communications*, pp. 1745–1753, 2011.

[13] M. Tripunitara and B. Carbunar, Efficient access enforcement in distributed role-based access control (RBAC) deployments, *Proceedings of the Fourteenth ACM Symposium on Access Control Models and Technologies*, pp. 155–164, 2009.