Lightweight Failure Detection in Secure Group Communication*

Patrick McDaniel† and Atul Prakash
Electrical Engineering and Computer Science Department
University of Michigan, Ann Arbor
pdmcdan{aprakash}@eecs.umich.edu

Abstract

The secure and efficient detection of process failures is an essential requirement of many distributed systems. In this paper, we present the design and analysis of a mechanism used for the detection of member failures in secure groups. Based on *one-time passwords*, our solution does not obviate the need for periodic statements from group members, but significantly reduces the cost of their generation and validation. A study comparing the costs of traditional mechanisms with our proposed approach is presented. Results of the study indicate the average case performance of the proposed scheme is $1/10th$ of traditional failure detection in trusted groups, and negligible in the untrusted groups. A discussion of security and performance tradeoffs made through mechanism policy is provided.

1 Introduction

As the Internet grows, the importance of group communication as an efficient means of distributing application content will increase. However, the security afforded by existing group oriented applications is insufficient to meet the needs of all session environments. Providing flexible, robust, and scalable security services within groups is an active area of research. A key requirement of these services is resiliency to failures of group participants. In this paper we present an efficient approach to one part of this resiliency; the secure detection of process failures.

A secure group is a collection of collaborating processes transferring content over a secured broadcast medium. Established via a group key distribution protocol [1, 2, 3, 4], a security context contains a *session key* known by each participant[1]. The group ensures the confidentiality, integrity, and authenticity of individual messages using the session key. However, there is an inherent cost to providing these guarantees.

*This work is supported in part under the National Science Foundation Grant #ATM-9873025 and by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0508. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

†Patrick McDaniel's research was supported in part by the NASA Graduate Student Researchers Program, Kennedy Space Center, Grant Number 10-52613.

[1]Groups whose members do not share a session key typically emulate a broadcast channel using secure point to point messages (e.g. the RAMPART system [5]). For simplicity, we assume groups share a common session key. However, our approach is also applicable to these types of groups.

Security is achieved by the cryptographic transformation of group messages using keys present in the security context. Due to the cost of these operations, it is desirable to limit the amount of data being transformed by the cryptographic algorithms.

Traditionally, detection of failed process requires the encryption and transmission of a periodic freshness indicator and identifier information under a known key. The freshness indicators may take the form of timestamps (where some secure source of time is globally available) or nonces. Typically known as *heartbeats*, the messages containing this information is generated by any party wishing to state its continued presence in the group, and validated by interested members. In high throughput, dynamic, or large groups, the costs of processing heartbeat messages from each member can be prohibitive.

In this paper, we present the design and analysis of an approach for the secure detection of failures. This approach amortizes the cost of heartbeat processing over many messages. Initially intended to be developed within the Antigone framework [6], the failure detection service can be implemented by any secure group having access to shared keying material. We also present a variant of our approach appropriate for groups with stronger security requirements. Based on public key algorithms, this variant is used in groups where compromised members may actively attempt to circumvent the failure detection mechanism. We consider the performance tradeoffs between the two variants in Section 3.

Based on *one-time passwords* [7, 8], our solution does not obviate the need for periodic heartbeat messages, but significantly reduces the cost of their generation and validation. Validation information is securely distributed to each member prior to the monitoring process. The validation information seeds an algorithm generating periodic secure heartbeats. These heartbeats are subsequently released by the member as proof of its continued presence in the group. Each heartbeat contains enough information to assess the freshness and authenticity of the message. Thus, we need not ensure reliable delivery.

Without loss of generality, we assume the group contains a single distinct member (called the *failure monitor*) monitoring the state of group members. We denote a member that has not failed as being *live*, and a member that has failed as being *failed*. We assume all failures are *fail-stop* [9]; a failed member will immediately and permanently stop transmitting messages to the group. We assume in mounting an attack, non-members (called *adversaries*) may attempt to intercept messages, modify messages, or prevent messages from being delivered.

The way in which groups should recover from failures is largely dependent on the the group threat model and session context. Several possible ways in which the group can react to the detection of a member failure include: a) the group disbands and suspends operation until the group member recovers and re-joins the group (in the case where is the failed process is essential to the group mission), b) the group can purge the member and continue, or c) ignore the failure. A thorough discussion of group failure recovery is outside the scope of this document.

The following section details the background and design of our proposed failure protection mechanism. Section 3 compares the cost of our proposed solution with traditional failure detection mechanisms. Section 4 gives a brief overview of the related work. Section 5 considers a number of issues relating to the integration and use of the proposed approach. We conclude in Section 6.

2 Secure Failure Detection

A failure monitor requires periodic proof that each monitored process has not failed. This proof typically takes the form of a member generated statement indicating its continued presence. These statements must be authenticated for the failure detection mechanism to be secure; some information proving the message was generated by stated member is necessary. If authenticating information is not included, an adversary can mask failures by generating counterfeit statements. In groups where the members are not completely trusted (or resiliency to member compromise is required), the authentication information must uniquely identify the

Notation	Description
A	The identity of a group member.
S	The identity of the failure monitor.
g	An identifier used to uniquely identify the (group) session.
$x_1, x_2, ..., x_n$	Concatenation of fields x_1 through x_n.
$H[x_1, x_2, ..., x_n]_k$	HMAC computation over fields $x_1, ...x_n$ with key k.
C_A	Digital certificate of member A.
$\{X\}_{C_A}$	Digital signature by A on X using private key associated with the certificate C_A.
k_g	The (symmetric) session key for group g.
S_A^0	The first heartbeat sequence number for member A.
S_A^i	The heartbeat sequence number i for member A.
δ^0	The seed value for a hash chain.
δ^k	The kth application of the hash function on an initial value of δ^0.

Table 1: Description of the notation used throughout.

sender. The ability to securely identify the sender of a group message, called *source authentication*, is an active area of research. Current methods providing source authentication can be prohibitively expensive or unreliable in high throughput groups [10, 11, 12].

Traditional mechanisms detect failures by the absence of periodic statements made by monitored processes. These statements, called heartbeats, are generated by each member and transmitted to the group directly or to the failure monitor via unicast. Each heartbeat must contain member identifying information and proof of its freshness. If a correct heartbeat is not received by the failure monitor, the process is deemed failed.

We use a simplified version of the Antigone secure failure detection mechanism to illustrate this approach. The notation used for the remainder of this paper is described in Table 1. Denoted as traditional failure detection (or simply the traditional approach) throughout, each member of group g is assigned a unique identifier (A), obtains the group session key (k_g), and establishes a starting sequence number (S_A^0) prior to the detection of failures. At some agreed upon periodicity, A generates and transmits the following message:

$$g, A, S_A^i, H[G, A, S_A^i]_{k_g}$$

When a failure monitor receives the message, it validates the authenticity of the message via the HMAC ($H[(\ldots)]_{k_g}$). If the message is authentic and the identifiers and sequence numbers are consistent with the expected values, the process is deemed live. The sequence number of the first heartbeat is S_A^0. Each subsequent heartbeat is incremented by one (i.e. $S_A^1, S_A^2, \ldots$).

When a member joins the group, a timer set to expire at the time the first heartbeat is expected is created. This timer is reset each time a correct heartbeat is received from the member. An expired timer indicates to the failure monitor that the process is no longer operating correctly. In this case, the failure monitor notifies the group that the process has failed and the group reacts appropriately.

We assume there is unique session identifying information present in the group identifier or session keys. For example, group identifiers in Antigone consist of a group description string and a nonce used to uniquely identify the session key. If identifiers and keys are reused between sessions, heartbeat messages can be replayed. However, in practice, it is unlikely session keys will be used for more than one session.

The failure monitor should allow additional time for network delays and lost messages when selecting the timeout value. Note that it may be acceptable to lose some heartbeats entirely. In accepting some

heartbeat loss, a failure monitor may tradeoff detection latency with resiliency to network packet drops. A detailed discussion of the selection of timer values is presented in [6].

The following subsections outline an approach that reduces the cost of heartbeat generation and validation over the traditional approach. This scheme modifies the way in which authentication is performed. In all other respects, the behavior of the monitored process and members is the same as in the traditional approach.

2.1 Approach

We begin the description of our approach by introducing the use of hash chains in authentication schemes. A hash chain [13] is the sequence of values resulting from the repeated application of a secure hash function (f) on some initial value. For example, given an initial value x, a one-way hash function $f(x)$, and chain of length $k+1$, the hash chain is:

$$f^0(x) = x, f^1(x), f^2(x), \ldots, f^k(x)$$

Because, by definition, (even partial) inversion of f is not feasible, knowledge of $f^i(x)$ gives no meaningful information to derive $f^{i-1}(x)$, for some $i, 0 < i < k$ [14]. By revealing $f^k(x)$ securely, the remaining values can be used in reverse order as proof of the knowledge of x. This is useful in authentication schemes (one-time passwords) because only a entity with knowledge of x can generate the intermediate values.

Once a value $f^i(x)$ has been used, an adversary can produce not only $f^i(x)$, but all $f^j(x)$ such that $j > i$. Thus, the values of the hash chain can only be used once and in reverse order. Because $f^i(x)$ is used only once, it can be sent in the clear. With slight modification, this approach is used in the S-Key authentication [7] system to authenticate users in distributed environments.

We now present our approach. We assume a session key shared by the members of the group and the failure monitor has been established prior to the detection of failures. For example, the GKMP protocol [1, 15] could be used to establish the group session key. Trivially, we also assume the failure monitor has some means of determining which process it is to be monitoring.

As in traditional schemes, we assume all processes have access to a hardware or software clock. The clock at each member need not be synchronized, but should not advance at significantly different rates. The proposed method can detect failures at rates bounded only by network latency and throughput. So, for the types of groups envisioned, it is unlikely any two clocks will progress at rates such that the correctness of the failure detection mechanism will be affected. We note some applications, such as real-time systems, may implement groups with very specific failure detection timing constraints. We defer a rigorous analysis of the timing requirements and constraints of these systems to future work.

Process failures are detected at the failure monitor by the absence of received heartbeat messages. Prior to the monitoring process, the member and the failure monitor must agree on an operating policy. In determining policy, each group member must state or accept the periodicity with which heartbeat messages will be transmitted. A second policy states the number of messages that may be lost before the member is deemed failed. In lossy environments, it may be desirable for members to frequently send heartbeats, and the monitor accept several consecutive losses before declaring the process failed. However, as the number of acceptable losses before failure detection increases, so does the latency with which failures are detected. Similarly, as the number of acceptable losses decreases, the probability that some live process is declared failed increases (false-positives). It is up to the application to identify parameters appropriate for its operating environment. The means by which participants agree on and distribute policy is outside the scope of this paper.

Based on policy, each member periodically transmits a *secure heartbeat* to the failure monitor. The freshness of each heartbeat message is indicated by a monotonically increasing sequence number. When

initiating the monitoring process, both the member and the failure monitor set the next/expected sequence number to zero. The member generates each heartbeat by inserting the next sequence number and incrementing the local sequence number counter by one (see below). At the failure monitor, a heartbeat containing the (authenticated) next sequence number is deemed valid and the expected sequence number is incremented by one.

Secure heartbeats are generated as follows. Initially, the member A generates a random value x of length equal to the output of the hash function (e.g. MD5 has a 128 bit output). A applies the hash function a member-determined number of times (k) to x to generate the following hash chain[2]:

$$\delta^0 = x, \delta^1 = f(x), \delta^2 = f^2(x), \ldots, \delta^k = f^k(x)$$

A generates a *heartbeat validation block* containing the group identifier g, her identity A, the first heartbeat sequence number for which this hash is to be used S_A^0, the last value in the hash chain $f^k(x) = \delta^k$, and a HMAC [16] covering these fields:

$$g, A, S_A^0, \delta^k, H[g, A, S_A^0, \delta^k]_{k_g}$$

A heartbeat message is generated by concatenating the current sequence number ($S_A^i = S_A^0 + i$) and the next value in the hash chain (in reverse order, δ^{k-i}) with the validation block. So, the transmitted heartbeat message for sequence number i would be:

$$S_A^i, \delta^{k-i}, g, A, S_A^0, \delta^k, H[g, A, S_A^0, \delta^k]_{k_g}$$

Because encryption is only required when creating the validation block and the hash chain itself is cached, heartbeat generation is fast. When the values of a chain are exhausted ($i > k$), the member can generates a new hash chain and associated validation block.

A received heartbeat message is validated by checking the HMAC and calculating:

$$f^{S_A^i - S_A^0}(\delta^{k-i}) = \delta^k.$$

If this relation holds, then the heartbeat is valid. The heartbeat is authentic because of the use of the shared secret key (or group private key) in the validation block. The heartbeat is fresh because of the presence of the next value in the hash chain. After receiving and validating the initial heartbeat for a hash chain, subsequent HMAC validation can be achieved by byte comparison of a validation block of a previously validated heartbeat. Thus, heartbeat validation is fast.

Each heartbeat is self-contained; all information needed to determine authenticity and freshness is present in the message. Therefore, the heartbeat messages need not be reliably sent. This scheme does not require the validation block be sent in the same message as any particular heartbeat. Acceptance only requires the validation block be obtained prior to validation. If some reliability service is used to deliver the validation block, one may decrease the transmission costs associated with failure detection by omitting the validation block from each heartbeat message.

As in our description of the traditional approach, we assume the group identifier g has enough information to uniquely identify the session key and group. However, in the case where this is not possible, replay protection can be achieved by introducing additional timestamp or nonce fields into the authentication block.

Any entity with access to the group session key can mask the failure of group members. Because the knowledge of the session key is used for authentication, the entity can generate heartbeats for other members. We address this limitation in a variant of hash chained heartbeats introduced in the following subsection.

[2]We note it may be desirable for the failure monitor to control the length of the hash chains. In these environments, the value can be negotiated or distributed in a manner similar to the other algorithm parameters.

	Traditional		Hash chained	
	Symmetric	Public Key	Symmetric	Public Key
Generation (avg, clock ticks)	6,650	8,584,340	2,160	818,290
Validation (avg, clock ticks)	5,810	51,890	2,590	51,890
Message Size (bytes)	52	164	86	198

Table 2: Summary of performance experiments. Hash chains of length 10 were used in all tests. If chains of longer lengths were used, improvements over the traditional approach would increase.

2.2 Failure Detection with Source Authentication

Some tightly coupled groups require each member to detect failures of all other members. This is useful in groups where distributed consensus is reached using voting protocols [5], where the group is sensitive to network partitions [17], or where group members are marginally trusted (i.e. the group wishes to be protected from the masking of failures by compromised members).

We use public key certificates to generate heartbeat messages and broadcasting to support failure protection in tightly coupled groups. Heartbeat message and hash chains are generated as described above, save that the HMAC is replaced by a digital signature generated using the private key associated with the public key certificate of the transmitting member. The resulting heartbeat for member A would be:

$$S_A^i, \delta^{k-i}, \underline{g, A, S_A^0, \delta^k, \{g, A, S_A^0, \delta^k\}_{C_A}}$$

Validation is performed by acquiring and validating the public key certificate of the sending member (if it is not cached), validating the digital signature, and checking the heartbeat relation as described above. Similar to the previous approach, after the first heartbeat message of a hash chain is validated, the heartbeat validation block (underlined portion of the above message description) can be validated by byte comparison.

This solution has a number of additional costs and requirements over the previously described (symmetric key) failure detection mechanism. First, the public key certificate of each member must be available and verifiably authentic. This requires the presence of some globally trusted certificate service (e.g. PKI). Due to issues of trust and performance, providing certificate services to members of groups spanning multiple administrative domains may be problematic [18].

A second cost of this scheme is the additional computational resources required to generate and validate the heartbeat messages. Public key algorithms can be up to 1000 times slower than symmetric key algorithms [19]. Thus, in large groups, the computation costs of processing these messages may be increased. However, because the digital signatures of each heartbeat message following the first can be validated by byte comparison, these costs are significantly lower than in traditional public key supported failure detection. We study the costs of both variants of this approach in the next session.

3 Performance

In this section, we compare the performance of our proposed approach with traditional failure detection. These experiments study the cost of heartbeat message generation and validation. We look at these costs in both symmetric key and public key based approaches. A summary of the experimental results described in this section is presented in Table 2.

Network latency and drop rates may affect the performance and efficiency of failure detection mechanisms. While both network latency and message loss may introduce *false failures*, these failures occur in either approach; a lost message in traditional failure protection will have exactly the same effect in the

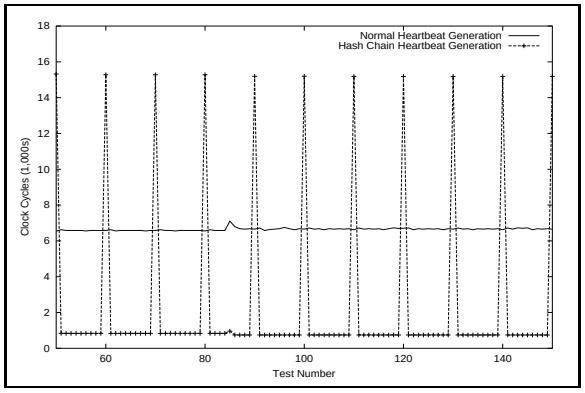

Figure 1: Symmetric key heartbeat generation costs - Cost of generating heartbeat messages in traditional and hash chained approaches.

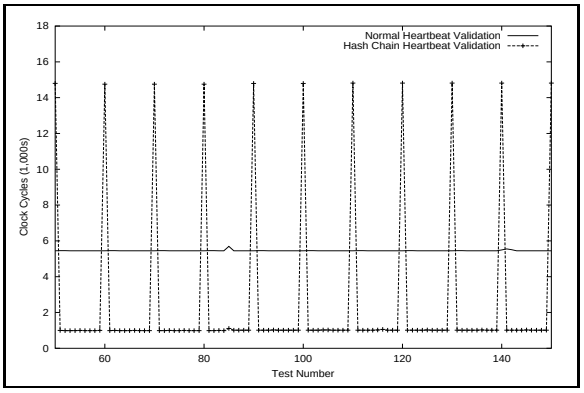

Figure 2: Symmetric key heartbeat validation costs - Cost of validating heartbeat messages in traditional and hash chained approaches.

approaches proposed in this paper. For the purposes of this study, we ignore network reliability and latency issues.

All tests were performed on a Linux kernel 2.2.14-5.0 running on a 500Mhz Pentium II. To remove the effects of processor speed on the performance measurements, kernel-level clock ticks are used as the performance metric. The DES [20] and RSA [21] (1024 bit keys) algorithms were used for the symmetric and public key operations, respectively. MD5 [22] was used for all cryptographic hashing. Keyed message authentication codes (HMACs) were generated per the RFC 2104 specification [16]. Message generation and validation methods where built on an alpha version of the Antigone 2.0 applications programmer interface libraries. The interfaces provided by the OpenSSL library version 0.9.5a [23] (upon which Antigone 2.0 is built) were used to implement all cryptographic algorithms and certificate processing. All hash chains contained 10 hash values.

Traditional failure detection is modeled through the generation and validation of signed or HMACed heartbeat messages. As described in section 2, traditional heartbeat messages contain a sequence number, a user identifier, a group identifier, and a HMAC computed over the previous fields. Upon reception of a heartbeat, the failure monitor validated the authentication information using the appropriate key or certificate, checked the user and group identifiers, and compared the sequence number against the expected value. The hash chained message format, generation, and validation is as described in the previous section.

In the first series of tests, we compare the cost of heartbeat generation in a symmetric key environment. Described in Figure 1, each tenth heartbeat costs 3 times as much as in the traditional approach[3]. These costs are due to the generation of a the new hash chain. Generation of the new hash chain requires the acquisition of random data, allocation of the appropriate structures, the hashing of the intermediate values, and the generation of the message itself. The costs of the traditional heartbeat are the construction of the message, the hashing of the message data, and the encryption of the hash data. Once the hash chain has been constructed, the costs of generating the validation block are roughly equivalent to the generation of a traditional heartbeat.

The generation of heartbeats using existing hash chain values is about $1/10th$ the cost of traditional generation. No hashing or encryption is required; the entire message can be generated by concatenating known values. Performance improvements of hash chained failure detection will increase as the length of the the hash chain increases. The cost of lengthening a hash chain by one value is negligible. However, because one may need to generate new hash chains based on reasons other than its values being exhausted,

[3]Each data-point in Figures 1, 2, 3, and 4 represents the time required to generate (or validate) a single message.

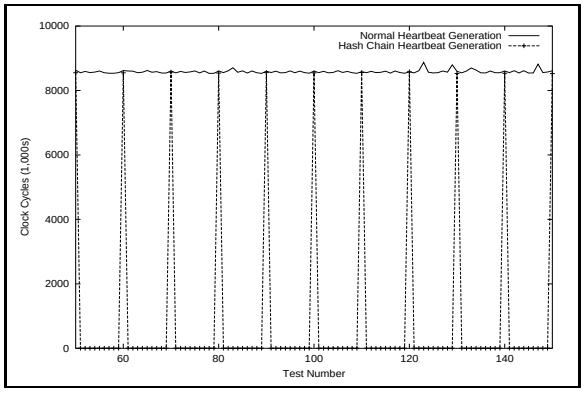

Figure 3: Public key heartbeat generation costs - Cost of generating heartbeat messages in traditional and hash chained approaches.

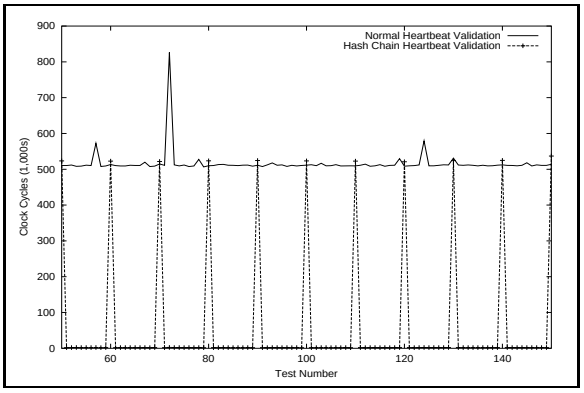

Figure 4: Public key heartbeat validation costs - Cost of validating heartbeat messages in traditional and hash chained approaches. Due to the performance of the underlying cryptographic algorithms, heartbeat generation is more than an order of magnitude slower than public key based validation.

it is often desirable to limit their length. We discuss factors contributing to the length of the hash chain in Section 5.

The performance of validation in the symmetric key environment mirrors generation. Described in Figure 2, both schemes reverse the generation process by extracting the appropriate values from the messages and validating the HMAC. When the first value from a hash chain is received, a new hash chain structure with the appropriate size is allocated, the validator is stored, and the validation block is validated. Subsequent validation of heartbeats created from the hash chain only require the application of the hash function on the received hash chain value. Thus, the cost of validation in the hash chained approach is about $1/7th$ of traditional failure detection.

The amount of state and hash processing by the failure monitor in the current implementation can be significantly reduced by storing only the last received value and the validation block. We expect this optimization would allow the performance of heartbeat validation with newly received hash chains to approach the cost of the traditional approach.

The advantages of our approach are more pronounced in the public key variant. For the reasons cited above, the hash chain costs are high at every tenth heartbeat, but relatively low for all other heartbeats. Described in Figures 3 and 4 and as compared to traditional failure detection, the additional costs of hash chain generation and processing are negligible. This is due in large part to the performance characteristics of the underlying cryptographic algorithms. The modular exponentiation required by RSA is computationally expensive. Thus, the allocation and initialization of hash chains is completely dominated by the cost of signing the validation block. As shown in figures 3 and 4, the cost of heartbeat generation and validation for the first value in a hash chain is roughly equivalent the traditional approach. These costs are negligible in all cases where a new validation block is not generated or authenticated.

In the public key variant, the cost of generation is an order of magnitude larger than validation. This is a desirable side-effect of the way in which RSA based certificates are generated. Dependent on the selection of public key exponents, operations using a private key (generation) can be significantly more expensive than operations using the public key (validation). This is also consistent with the needs of the failure detection. The failure monitor may need to process (validate) many heartbeats. However, each member need only generate one heart per period.

An advantage of a traditional approach over hash chained failure detection is message size. Described in table 2, heartbeats based on hash chains are slightly larger than traditional heartbeat messages. However, because of the frequency of these messages, it may not likely to effect session throughput.

A second limitation of this approach is the amount of additional state needed at both the sender and failure monitor. To obtain the performance advantages of this approach, the sender needs to store the current hash chain intermediate values. However, they may be recalculated as necessary without significantly reducing the generation costs. Similarly, the failure monitor needs to keep track of the latest authentication information (authentication block). If this information is not stored, the monitor will not receive the benefit of past authentication. The amount of state held at each process can be significantly reduced by storing only the most recently received values. In the current implementation, a failure monitor (sender) could store as little as 214 (110) bytes (198 (94) authentication block + 16 last received (sent) hash value) and still obtain the reported performance.

4 Related Work

Many of the existing techniques used for the detection of failures in group communication systems were developed within the context of reliable multicast. Often cited as the genesis of reliable group communication, the Isis [24] and later HORUS [25] group communication platforms developed many of the mechanisms under which contemporary reliable group communication systems are built. The robust detection of failures is often essential to these mechanisms. Typically, reliable multicast frameworks detect failures by the presence or absence of message acknowledgements. A central property of reliable multicast is *safety*; a message is either received by every non-failed member or by none. Thus, systems providing safety much receive an acknowledgement from each member before committing it. Because these acknowledgements implicitly state a non-failed state, they serve a dual purpose as messages acknowledgements and member heartbeats. More recent systems (i.e. TRANSIS [17], Ensamble [26], and CACTUS [27]) rely on similar reliability or *process group management* protocol mechanisms to detect failures.

The RAMPART system [5] provides secure group communication in the presence of actively malicious processes and Byzantine failures. Protocols in RAMPART rely heavily on distributed consensus algorithms to reach agreement on the course of group action. Secure channels between pairs of members are used to ensure message authenticity. Authenticity guarantees are used to ensure the accuracy of the group information and for the detection of failures. To simplify, members of RAMPART groups are removed (detected as failed) from the current group if they are deemed unresponsive (i.e. fail to respond to group messages).

Similar to the traditional heartbeats, group members in the Iolus system [28] are required to periodically re-assert their presence in the group. However, rather than using these messages as periodic proof of a process's continued presence, they provide a means by which a group member can REFRESH its membership in the group. Each member is assigned a membership expiration time during the join process. The member may refresh its membership prior to the expiration time. Any member who does not refresh its membership before the expiration time is ejected from the group. While this approach has the advantage of being sender driven, it requires a refresh acknowledgement mechanism. Without an acknowledgement, the refresh can be lost and the member incorrectly ejected from the group.

The Antigone framework [6] provides flexible interfaces for the definition and implementation of a wide range of secure group policies. A central element of the Antigone architecture is a set of *mechanisms* providing the basic services needed for secure groups. The Antigone failure detection and recovery mechanism defines an abstract interface for the detection of member failures and compromises. In its initial version, each Antigone process transmits a periodic heartbeat message asserting their continued presence in the group. The definition of the traditional failure detection used throughout this paper was taken directly from this work.

The security requirements of failure detection mechanisms are similar to those of source authentication in groups. Each facility requires the sender to provide authentication information for transmitted data. While general purpose source authentication requires the sender to commit to some previously unknown data (the contents of a packet), failure protection only requires the sender generate an authenticated message containing a previously and globally known sequence number. Thus, in failure protection schemes, the sender can use pre-authenticated information (in our proposal, the intermediate values of the hash chain). Any mechanism providing source authentication can be used for failure detection. Furthermore, our approach can be augmented with any of these schemes; rather than encrypting or signing the authentication block, one could use any existing source authentication mechanism.

Perrig et. al. [10] define an efficient mechanism providing source authentication in groups. This approach is based on sender commitment to an authentication key during transmission (at time t) which is later exposed (at time $t + \delta$). If a receiver can prove the message was received before the key was exposed (message received $< t + \delta$), then it is deemed authentic. Other approaches for achieving source authentication in groups are described in [11, 12, 29].

5 Discussion

The length of the hash chain used in this scheme will influence the efficiency of failure detection. If the chain is long, the costly heartbeat validation block generation process will occur infrequently. However, if other factors require the hash chain be discarded, the unused intermediate values represent wasted effort. For example, failure monitor may desire newly received session keys be used to generate heartbeats. In this case, each receiver must establish a new hash chain and associated validation block after receiving the new key. Thus, when deciding the length of a hash chain, the process must take into account the length of time over which it is likely to be useful.

An interesting question is whether secure heartbeats may be used for other purposes. For example, in Antigone [6], heartbeats are not only used to detect failures, but also as implicit acknowledgements of group keying material. If each heartbeat validation block includes session key identifiers, then it can also be used as a receipt of the most recent key distribution message. A similar approach may be used to reduce the cost of secure acknowledgement (or negative acknowledgement) generation in reliable group communication.

One optimization of this approach is to piggyback secure heartbeats onto other group messages. For example, a secure heartbeat could be concatenated onto each key distribution or data message. This is similar to the optimization of HORUS [30], where potentially many data messages are compressed (*packed*) and sent in a single transmission.

In large groups, requiring each member to generate and transmit secure heartbeats may be infeasible. A potential modification to our approach addressing this limitation would be the set the frequency with which members are required to transmit heartbeats independently. Members essential to the group (e.g. senders, arbitrators) would transmit heartbeats more frequently than non-essential members. Non-essential members would transmit heartbeats less frequently or not at all.

6 Conclusions

In this paper we have presented the design and analysis of an approach for the secure and efficient detection of group member failures. We reduce the costs associated with traditional failure detection by using *one-time-passwords* as proof of members' continued correct operation. This approach amortizes the costs of secure heartbeat generation and validation by distributing the initial value of an authenticated hash chain. Subsequent heartbeats are generated using the hash chain intermediate values as freshness indicators. By maintaining state at monitoring processes, we reduce these costs without sacrificing correctness or security.

The performance study shows the proposed approach can significantly reduce the message processing costs of traditional failure detection mechanisms. In trusted environments, the cost of heartbeat processing is roughly $1/10th$ the cost of traditional heartbeat processing in the average case. In the exceptional case where a new hash chain is generated, our approach is about 3 times as expensive. However, through manipulation of the of the hash chain, the frequency with which the exceptional case occurs is in control of the sending process.

In environments where group members are not completely trusted or resiliency to member compromise is required, the performance of the hash chained approach is roughly equivalent to traditional approaches in the worst case, and negligible in the average case. Thus, for these environments, the secure detection of failures can scale to very large groups.

Our solution addresses one problem found facing secure groups; the reliable and secure detection of failures. Other outstanding issues include recovery from member failures, scalability of failure detection, and the effects of failed members on the group security context. We plan to investigate these and other issues in the future.

7 Acknowledgements

We would like to thank Avi Rubin for his invaluable advice and analysis on the design of this approach.

References

[1] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Specification. *Internet Engineering Task Force*, July 1997. RFC 2093.

[2] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *International Conference on Distributed Computing Systems (ICDCS'98)*. IEEE, May 1998.

[3] Bob Briscoe. MARKS: Zero Side-Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences. In *Proceedings of First International Workshop on Networked Group Communication*, November 1999.

[4] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communication Using Key Graphs. In *Proceedings of ACM SIGCOMM '98*, pages 68–79. ACM, September 1998.

[5] M. Reiter. Secure Agreement Protocols: Reliable and Atomic Group Multicast in Rampart. In *Proceedings of 2nd ACM Conference on Computer and Communications Security*, pages 68–80. ACM, November 1994.

[6] P. McDaniel, A. Prakash, and P. Honeyman. Antigone: A Flexible Framework for Secure Group Communication. In *Proceedings of 8th USENIX UNIX Security Symposium*, pages 99–114. USENIX Association, August 1999. Washington D. C.

[7] N.M. Haller. The S/Key^{tm} One-Time Password System. In *Proceedings of 1994 Internet Society Symposium on Network and Distributed System Security*, pages 151–157, February 1994. San Diego, CA.

[8] Aviel D. Rubin. Independent One-Time Passwords. *USENIX Journal of Computer Systems*, 9(1):15–27, February 1996.

[9] Sape Mullender. *Distributed Systems*. Addison-Wesley, First edition, 1993.

[10] Adrian Perrig, Dawn Song, Doug Tygar, and Ran Canetti. Efficient Authentication and Signature of Multicast Streams over Lossy Channels. In *2000 IEEE Symposium on Security and Privacy*, page (to appear). IEEE, May 2000. Oakland, CA,.

[11] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. In *Proceedings of CRYPTO 97*, pages 180–197, August 1997. Santa Barbara, CA.

[12] C. Wong and S. Lam. Digital Signatures for Flows and Multicasts. In *6th International Conference on Network Protocols*. IEEE, 1998. Austin TX.

[13] Leslie Lamport. Password Authentication with Insecure Communication. *Commuications of the ACM*, 24(11):770–772, 1981.

[14] D. Stinson. *Cryptogaphy: Theory and Practice*. CRC Press, first edition, 1995.

[15] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Architecture. *Internet Engineering Task Force*, July 1997. RFC 2094.

[16] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. *Internet Engineering Task Force*, April 1997. RFC 2104.

[17] D. Dolev and D. Malki. The Transis Approach to High Availibility Cluster Communication. *Communications of the ACM*, 39(4), April 1996.

[18] P. McDaniel and A. Rubin. A Response to 'Can We Eliminate Certificate Revocation Lists?'. In *Proceedings of Financial Cryptography 2000*. International Financial Cryptography Association (IFCA), February 2000. Anguilla, British West Indies. (to appear).

[19] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc., second edition, 1996.

[20] National Bureau of Standards. Data Encryption Standard. *Federal Information Processing Standards Publication*, 1977.

[21] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[22] R. Rivest. The MD5 Message Digest Algorithm. *Internet Engineering Task Force*, April 1992. RFC 1321.

[23] The OpenSSL Group. OpenSLL, May 2000. URL: `http://http://www.openssl.org/`.

[24] K. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of the ACM*, 36(12):37–53, December 1993.

[25] R. Van Renesse, K. Birman, and S. Maffeis. Horus: A Flexible Group Communication System. *Communications of the ACM*, 39(4):76–83, April 1996.

[26] O. Rodeh, K. Birman, M. Hayden, Z. Xiao, and D. Dolev. Ensemble Security. Technical Report TR98-1703, Cornell University, September 1998.

[27] Matti A. Hiltunen, Sumita Jaiprakash, and Richard D. Schlichting. Exploiting Fine-Grain Configurability for Secure Communication. Technical Report TR99-08, Department of Computer Science, University of Arizona, May 1999.

[28] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings of ACM SIGCOMM '97*, pages 277–278. ACM, September 1997.

[29] Pankaj Rohatgi. A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication. In *Proceedings of 6th ACM Computer and Communications Security Conference*. ACM, 1999. Singapore.

[30] R. Friedman and R. van Renesse. Packing Messages as a Tool for Boosting the Performance of Total Ordering Protocols. Technical Report TR-95-1527, Department of Computer Science, Cornell University, July 1995.