INTEGRATING PUBLIC KEY CRYPTOGRAPHY INTO THE SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP) FRAMEWORK

Tat Chee Wan (tcwan@cs.usm.my)[1]

[1]Network Research Group
School of Computer Science
University of Science Malaysia
11800 Penang, Malaysia
Tel: (04) 659-4757
Fax: (04) 657-3335

Alwyn Goh (alwyn@cs.usm.my)[2]
Chin Kiong Ng (ckng@cs.usm.my)[2]
Geong Sen Poh (gspoh@cs.usm.my) [2]

[2]Health Informatics Research Group
School of Computer Science
University of Science Malaysia
11800 Penang, Malaysia
Tel: (04) 657-4698
Fax: (04) 657-3335

ABSTRACT

***Keywords**: SNMP Agents, SNMPv3, Security Models, Public Key Cryptography.*

Simple Network Management Protocol (SNMP) is widely used for remote network resource management due to its simplicity and distributed management capabilities. However, the increased use of SNMP to manage and control network resources such as routers and servers also introduces security risks whereby unauthorized users can retrieve information or modify the given resources remotely. The basic security framework introduced in SNMPv3 only specifies the use of symmetric cryptography techniques to address the security concerns. This paper outlines a new methodology, Public-Key Security Model (PSM), to integrate public cryptography techniques into the SNMP framework. It extends the existing User-based Security Model (USM) to include per-session authentication and encryption keys, thus enhancing the security of the SNMPv3 protocol.

I. INTRODUCTION

Simple Network Management Protocol (SNMP) is an IETF standard protocol for remote management of network resources. In the SNMP architecture, *Agents* are software modules that reside on the network resource to provide information-reporting and configuration services to SNMP *Managers*. The SNMP architecture has undergone several revisions, from SNMPv1 that did not provide any access control and privacy for information retrieval and configuration, to SNMPv3 that provided basic authentication and encryption services using symmetric cryptography techniques.

II. SNMPV3 SECURITY MODELS

SNMPv3 defines a generic architecture to support network management functions in a modular manner [2]. An SNMP entity is defined either as agents or managers. Agent monitor specific performance metrics of networked devices and report them to manager entities. Each SNMP entity comprises of SNMP *Applications* that utilizes a SNMP *Engine* to exchange data and commands between managers and agents (Figure 1). Furthermore, the engine provides modules for *message dispatch*, *message processing*, *security* and *access control*. Each module within the SNMP engine is defined to be replaceable, to permit the updating and enhancements of individual modules independently of the others. The message dispatcher forwards messages received from the transport layer to the respective message processing modules (to cater for the various versions of SNMP protocols) [1].

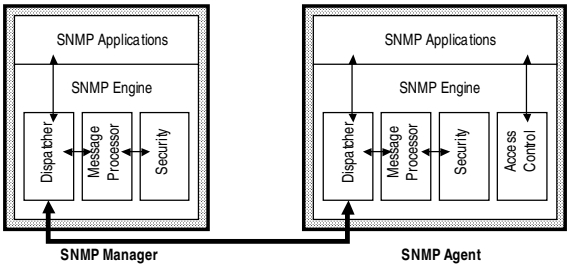

Figure 1: Generic SNMP Architecture Framework

Security issues can be categorized into Authentication, Encryption and Access Control. Within SNMPv3, the Security Module handles Authentication and Encryption, while the Access Control Module handles Access Control in agent entities. There is a separation of these functions since Authentication and Encryption are orthogonal to Access Control issues. SNMPv3 implements security by prepending a security header to a standard SNMP Protocol Data Unit (PDU), thus allowing entities to support non-SNMPv3 aware agents and managers. The focus of this paper is on Authentication and Encryption; we assume that the default Access Control methodology, View-based Access Control Model (VACM) [4] is sufficient for addressing access rights issues. Currently, only one security model, the User-based Security Model (USM) is defined [3]. SNMPv3 allows for unauthenticated and unencrypted messages, authenticated but unencrypted messages, and authenticated encrypted messages between managers and agents [1]. Nonetheless, the SNMPv3 standard leaves room for alternative implementations via the *msgSecurityModel* field in the SNMPv3 PDU.

The goal of any security model is to prevent at minimum, the following attacks [1]:

- *Modification of Information*: access and modification of SNMP entities by unauthorized entities via alteration of intercepted messages, e.g., "Man in the Middle" attacks
- *Masquerade*: Operations originating from unauthorized entities assuming the identity of a legitimate entity
- *Message Stream Modification*: delayed messages, message replay, message reordering to effect unauthorized management operations
- *Disclosure*: compromised SNMP message content

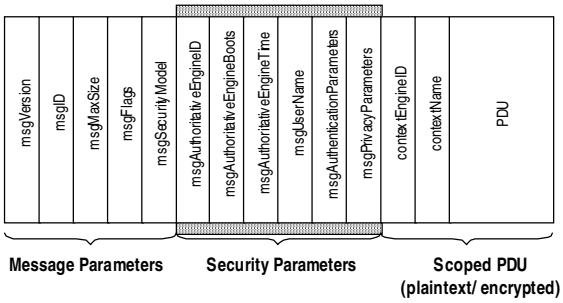

Figure 2: SNMP Message Format

This USM framework is used to authenticate entities, and provide encryption services to secure communications between agents and managers. The USM framework was designed to be distributed, where each agent keeps track of the authorized users and allowable access via an internal table of {user, password, access} entries. Both authentication and encryption (privacy) utilize symmetric keys generated from the stored passwords. Localization of the authentication and encryption keys is allowed by hashing the generated key with the ID of each agent entity. This prevents the compromise of all agents if an intruder manages to compromise the keys of a given agent. Authentication is performed using Hashed Message Authentication Code (HMAC) based on techniques such as Message Digest Version 5 (MD5) and Secure Hash Algorithm (SHA), while encryption is performed using Data Encryption Standard Cyclic Block Chaining Mode (DES-CBC) [1]. While this provides basic security features for SNMP, it is not sufficient in a distributed Internet environment where malicious users can perform a variety of attacks (e.g., via username and password interception, dictionary attacks, session hijacking, etc.) to gain unauthorized access to SNMP enabled resources.

Public Key Cryptography (PKC) is recognized as a highly secure encryption technique that avoids the issues inherent in symmetric cryptography. While SNMPv3 alludes to the feasibility of using PKC for securing manager-agent communications, it was left unspecified due to the need to define a basic working SNMP security model first and foremost. For historical reasons, security specifications that were not implemented in SNMPv2 became the basis for SNMPv3, which utilizes symmetrical cryptography techniques using fixed encryption keys. In contrast, PKC utilizes unique session keys for each transaction in place of the fixed encryption key. In addition, authentication using a variant of Diffie-Hellman (DH) key exchange technique [10], termed *Intertwined DH2 Key Negotiation* (IDH2), eliminates the weaknesses present in standard authentication techniques.

Nonetheless, PKC assumes the presence of a centralized Key Server, which differs from the assumptions made in the USM framework. This paper describes an approach whereby the centralized nature of PKC can be incorporated into the distributed USM framework, while still preserving the essential distributed nature of the SNMP protocols.

III. SECURE DIFFIE-HELLMAN PROTOCOL FOR AUTHENTICATION AND KEY-EXCHANGE

The Diffie-Hellman (DH) protocol is designed to allow a manager $\mathbf{E_M}$ and an agent $\mathbf{E_A}$ to securely exchange a session encryption key *K*. A session authentication key *K'* can also be generated at the same time, if desired. Each entity independently computes a Discrete Logarithmic (DL) parameter that can subsequently be bit-reduced via hash computation and used as the encryption key *K* for block-cipher encryption. DH assumes the availability of a shared DL environment (*p, g*), and requires $\mathbf{E_M}$ and $\mathbf{E_A}$ to generate the following session key pairs respectively: (x, $x' = g^x \bmod p$) and (y, $y' = g^y \bmod p$).

Correct execution by both parties results in synchronization onto the common DH parameter $\chi = g^{xy} \bmod p$, to generate the session encryption key $K = h_1(\chi)$, and the session authentication key $K' = h_2(\chi)$; thereby enabling computation and verification of zero-knowledge (ZK) proofs (π_a, π_m). This variant of the DH protocol involves the additional step of transmitting the ZK proof from the manager to the agent to ascertain successful completion of the DH protocol. Subsequent SNMP data is transmitted using the encryption key *K*.

The major problem arises from the choice of DL key pairs (x, x′) and (y, y′), which should be session-specific so as to ensure generation of fresh *K* and *K'*. Usage of long-term key pairs would defeat the purpose of this exercise. However, the featured session-specific key pairs cannot be authenticated as belonging to the participating entities. Basic DH is therefore vulnerable to a man-in-the-middle attack where malicious entity $\mathbf{E_X}$ interposes himself between $\mathbf{E_M}$ and $\mathbf{E_A}$, assuming $\mathbf{E_M}$'s identity to $\mathbf{E_A}$ and vice-versa.

This issue can be efficiently dealt with via incorporation of two key-pairs per entity, namely the use of long term key pairs (with the public keys stored in a PKC key server), and session specific key pairs generated for a given SNMP session [10]. This new protocol is termed *Intertwined DH2 Key Negotiation* (IDH2), which builds upon the two key-

pairs concept developed in Unified DH (DH2) [8],[9], to provide higher security using less protocol steps.

Long-term key pairs comprise of (private, public) key pairs generated *a priori* for both the Manager $\mathbf{E_M}$ (μ, $M = g^{\mu} \bmod p$) and Agent $\mathbf{E_A}$ (α, $A = g^{\alpha} \bmod p$) respectively. The public keys for each entity must be propagated via the PKC Key Server to all entities participating in a given transaction. Session-specific key pairs for the Manager $\mathbf{E_M}$ (x, x′) and Agent $\mathbf{E_A}$ (y, y′), are used as previously explained.

The use of two key pairs allows for entity authentication via proper execution of a challenge-response sequence designed to verify private-key knowledge, and eliminates the possibility of man-in-the-middle attacks by making it impossible for $\mathbf{E_X}$ to construct the correct ZK proofs. The resultant protocol, *Intertwined DH2 Key Negotiation* (IDH2) integrates entity *authentication* and *key negotiation* functionalities. However, it accomplishes the additional task of authentication without adding any extra protocol exchange steps to the DH2 protocol.

Synchronization onto $\chi = g^{(x+c_m\mu)(y+c_a\alpha)} \bmod p$, session encryption key $K = h_1(\chi)$, and session authentication key $K' = h_2(\chi)$ requires demonstrable knowledge of μ by $\mathbf{E_M}$, and α by $\mathbf{E_A}$, without which the ZK proofs cannot be correctly constructed. Such knowledge is equivalent to entity authentication, which in this case occurs as an integral part of the key-negotiation process.

It should be pointed out that entity authentication can actually be established quasi-independently of key-negotiation; i.e. via signature affixation on the session-specific public-keys x' and y'. This would however be somewhat less compute-efficient (depending on the signature protocol employed) than the above-described method, which effectively integrates both processes at the cost of one additional exponentiation per participating entity and introduced an extra parameter into the message exchange.

IV. IMPACT OF PUBLIC-KEY SECURITY MODEL (PSM) ON SNMPV3

Any attempts to incorporate PKC into SNMP must address the following issues: modification of information, masquerading, message stream modification and disclosure. In addition, several issues unique to PKC are addressed: centralized vs. distributed access controls, trusted vs. untrusted node authentication, as well as key maintenance and distribution. We proposed a new scheme, Public-Key Security Model (PSM) to address the limitations inherent in USM. PSM-based SNMP PDUs have a unique value for the *msgSecurityModel* field.

SNMPv3 treats the functions of authentication and encryption as separate issues. Consequently, existing authentication and encryption algorithms are replaced with stronger algorithms: Authentication is performed using SHA-based HMACs, while encryption is performed using International Data Encryption Algorithm (IDEA).

USM utilizes Localization of the security keys to provide some measure of protection against key compromise, since the compromise of one key does not enable the intruder to access other agents. In PSM, localization is not required, since private keys are not shared between managers and agents. Furthermore, entity authentication is inherent in PKC-based approaches, which provides protection against masquerading entities. In addition, the use of per session encryption and authentication keys reduce the danger posed by modification of information, masquerading, message stream modification and disclosure due to the limited applicability of a given key to a specific transaction session.

The per-session authentication and/or encryption keys are established by means of a new algorithm called *Intertwined DH2 Key Negotiation* (IDH2), which builds upon the two key-pair concept developed in Unified DH (DH2) [8],[9] to provide higher security using less protocol steps. Details of the algorithm are described in [10]. Instead of the implicit static authentication and encryption key generation used in USM, PSM establishes unique authentication and encryption keys for each transaction between agents and managers.

	Manager $\boldsymbol{M}$		*Agent* $\boldsymbol{A}$
1	Generate (x, x′) Generate C_a Transmit (x', C_a)	$\xrightarrow{(x', C_a)}$ $\xleftarrow{(y', C_m, \pi_a)}$	Generate (y, y′) Generate C_m Compute $\chi = \left(x' M^{c_m}\right)^{(y+c_a\alpha)} \bmod p$ Compute $K = h_1(\chi)$, $K' = h_2(\chi)$ Compute $\pi_a = h(x', y', K)$ Transmit (y', C_m, π_a)
2	Compute $\chi = \left(y' A^{c_a}\right)^{(x+c_m\mu)} \bmod p$ Compute $K = h_1(\chi)$, $K' = h_2(\chi)$ Verify π_a Compute $\pi_m = h(x', y', \pi_a, K)$ Transmit π_m	$\xrightarrow{\pi_m}$	Verify π_m

Figure 3: Algorithmic Description of Intertwined DH2 (IDH2) Key Negotiation

This is achieved by modifying the two-way handshake between the manager and agent in USM to become a three-way handshake in PSM at the transaction establishment phase. The protocol establishes the identity of both entities to each other (node authentication), session authentication key and session encryption key before any transactions occur.

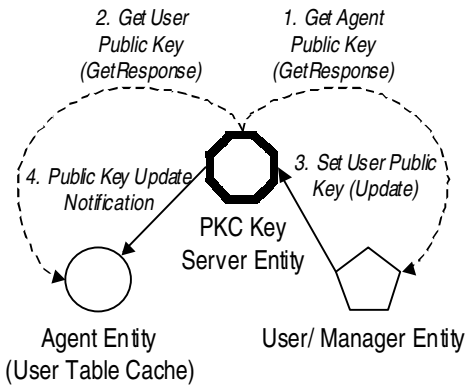

Figure 4: Role of PKC Key Server — provide public keys upon request (1 & 2), Maintain public keys (3), and Propagate updated keys (4)

Nonetheless, PSM introduces an additional component into the SNMP framework: a Public Key Cryptography Key Server (PKC Key Server) to distribute public keys belonging to respective entities (agents and manager users). The PKC Key Server could be implemented as an SNMP agent with a special MIB storing {username, Public Key, access} pairs. It would respond to get requests for relevant public keys for agents and users, as well as perform key propagation via Notifications when a command to update a Public Key is validated.

IV.1. Preventing Common Security Attacks

Modification of Information, in the form of intercepted messages via "Man in the Middle" attacks, is an inherent weakness in USM due to the use of shared symmetric authentication and privacy keys. PSM utilizes the IDH2 protocol to guard against such attacks via per-session keys [10]. In contrast, USM uses a fixed algorithm to generate privacy (encryption) keys for all transactions, making it much more vulnerable to external attacks. Since the PSM per-session keys are never transmitted to the receiver, there is little possibility of an intruder successfully masquerading as an authentic user. The presence of a PKC key-server further allows for trusted node authentication between the manager and agent, further reducing the risk of masquerading.

The USM implements a timeliness mechanism using the set of variables {*snmpEngineBoots*, *snmpEngineTime*, *lastReceivedEngineTime*} to guard against reordering and replay attacks [1]. PSM utilizes the same mechanism to enforce message ordering and transaction integrity, since the underlying UDP transport is unreliable. It is further augmented by the use of session authentication keys and session encryption keys that are generated for each transaction to prevent replay attacks. Since session keys are retired after a given timeout interval, and a new session key negotiated for future transactions, the likelihood of an intruder being able to compromise both the standard timeliness mechanism and the session encryption key simultaneously is significantly reduced. Even if the session keys were compromised, the risk of disclosure is reduced to that for a given transaction, compared with USM where once the privacy key is compromised, the intruder has the ability to access all future transactions.

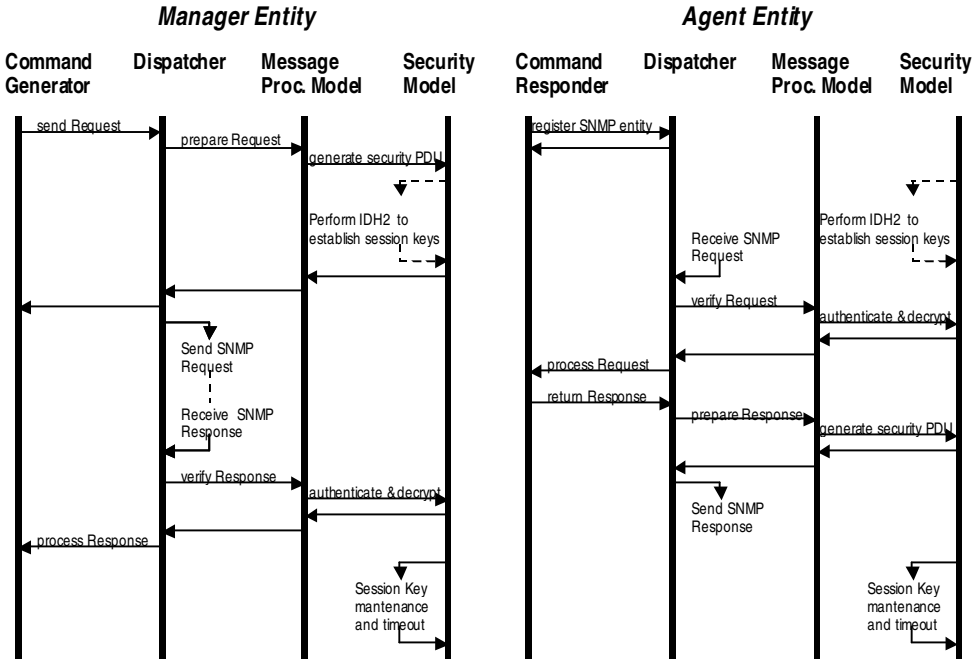

Figure 5: PKC-based Security Model Event Flows between Manager and Agent Entities

IV.2. Centralized vs. Distributed Access Control

PKC techniques necessitate the introduction of a Public Key Server into the security framework. The key server is used to store the public keys of each agent and each user who is part of the SNMP 'community' authorized to access SNMP-enabled network resources. This is in contrast to the current USM framework, where access to network resources are controlled on a per-resource basis using the {*username*, *password*, *access*} table stored in each SNMP agent. The advantages of the distributed USM framework are that it is much simpler, as well as not dependent on a single point of failure in the network, since each agent only need to perform the authentication and encryption steps with the corresponding manager.

In the proposed approach, we utilize the key server as a central database to keep the public keys, while at the same time, cache the {*username*, *public key*, *access*} entries into the USM table in place of the existing {*username*, *password*, *access*} configuration. This reduces the need to access the public key server to situations when the cached copies have become outdated. In addition, fine-grained access control is still supported under this scenario since each agent can be configured with a different list of users and access rights.

IV.3. Trusted vs. Untrusted Node Authentication

The strength of PKC based node authentication techniques is that it is trusted, since only the relevant parties would be able to successfully complete the node authentication and key negotiation process without disclosing their private keys. In contrast, USM requires that the shared secret key be safe from compromise when it is updated. While typical PKC based schemes require the presence of a Public Key Server in the transaction, adding additional communications overhead to the situation, the use of cached entries in the agents would eliminate the additional communications used to perform normal node authentication. The amount of additional traffic to the Public Key Server is reduced to a minimum, and is only caused by cache updates or explicit three way authentication requirements.

IV.4. PKC Public Key Maintenance and Distribution

PKC schemes typically specify a validity period for each PKC public key. Periodic public key updates are a prudent means to guard against compromise of the private key via cryptanalysis attacks. The entity must first transmit an authenticated public key update message to the PKC server to modify its public key. Subsequently, the PKC server will transmit public key update notifications to all SNMP agents to refresh their User Table Cache.

V. IMPLEMENTING PSM TRANSACTIONS

V.1. Agent Discovery

Agent Discovery is the process of synchronization between manager $\mathbf{E_M}$ and agent $\mathbf{E_A}$ entities for the purpose of establishing a SNMP transaction by configuring {*snmpEngineID, snmpEngineBoots, snmpEngineTime*} before any SNMP messages are exchanged, via two discovery transactions [5]. In addition, PSM performs IDH2 during the Agent Discovery Phase to establish the per-session Authentication Key *K'* as well as Encryption Key *K* as needed.

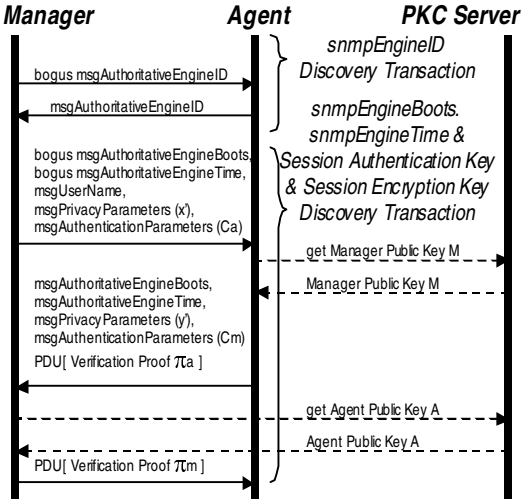

Figure 6: PSM Agent Discovery Phase Incorporating IDH2

- First *snmpEngineID* Discovery transaction (default):
  1. $\mathbf{E_M}$ sends an SNMPv3 packet with bogus *msgAuthoritativeEngineID* to Agent.
  2. $\mathbf{E_A}$ responds with the correct *msgAuthoritativeEngineID*.
- Second *snmpEngineBoots* & *snmpEngineTime* Discovery transaction, incorporating IDH2:
  1. $\mathbf{E_M}$ generates a Session Private Key x, a Session Public Key x', and a challenge C_a.
  2. $\mathbf{E_M}$ sends a PSM tagged SNMPv3 packet containing bogus *msgAuthoritativeEngineBoots*, *msgAuthoritativeEngineTime*, a valid Manager username (*msgUserName*), Session Public Key x' (*msgPrivacyParameters*) and the challenge C_a (*msgAuthenticationParameters*) to $\mathbf{E_A}$.
  3. $\mathbf{E_A}$ generates Session Private Key y, Session Public Key y' and challenge C_m.
  4. $\mathbf{E_A}$ uses the *msgUserName* to look up $\mathbf{E_M}$'s Public Key *M* in its User Table, and compute $\chi = \left(x' M^{c_m}\right)^{(y+c_a\alpha)} \bmod p$ and derive the tentative Session Encryption Key $K = h_1(\chi)$ and tentative Session Authentication Key $K' = h_2(\chi)$.

5. The correct *msgAuthoritativeEngineBoots*, *msgAuthoritativeEngineTime* values, $\mathbf{E_A}$'s session public key y' (*msgPrivacyParameters*), challenge C_m (*msgAuthenticationParameters*) and computed verification proof π_a stored in the PDU data field, are returned to $\mathbf{E_M}$.

6. An error is returned if $\mathbf{E_A}$ does not support PSM. If $\mathbf{E_M}$'s Public Key *M* is not cached in the Agent's User Table, it is retrieved from the PSM Key Server via appropriate transactions.

7. $\mathbf{E_M}$ computes $\chi = \left(y' A^{c_a}\right)^{(x+c_m \mu)} \bmod p$ using $\mathbf{E_A}$'s Public Key *A*, derive the Session Encryption Key $K = h_1(\chi)$ and Session Authentication Key $K' = h_2(\chi)$, as well as authenticates the response using π_a. Once verified, $\mathbf{E_A}$'s *snmpEngineBoots* & *snmpEngineTime* are used.

8. $\mathbf{E_M}$ computes its verification proof π_m and transmits it to $\mathbf{E_A}$ as PDU data in the reply packet.

9. Upon verification of π_m, $\mathbf{E_A}$ accepts *K* and *K'*, completing the Agent Discovery phase.

V.2. Authentication and Privacy

PSM utilizes the standard HMAC algorithms (HMAC-MD5 or HMAC-SHA) to perform authentication using the per-session Authentication Key *K'*. The HMAC value is stored in *msgAuthenticationParameters* for all subsequent data messages. If required, privacy in PSM is provided via the per-session Encryption Key *K* using standard encryption algorithms provided in USM. However, PSM must maintain the state of the session via additional status parameters to account for key expiry and transaction termination. Since the keys are to be invalidated after the completion of a transaction, the manager entity would transmit a set command, specifying the termination of the transaction. When this occurs, the manager and agent entities must perform a new DH session key exchange when a new transaction is required. Alternatively, the timeout mechanism prevents stale session keys from being retained, if an abnormal session termination occurred.

V.3. Session Key Maintenance

The session keys are active for the duration of the transaction. A key expiration timer is reset upon receipt of each SNMP message. Upon expiry of the timer due to inactivity, new session keys must be negotiated. In addition, the manager can force the invalidation of session keys by sending an empty PDU with the encryption and authentication flag disabled after the last message exchange is complete to signal the termination of the transaction.

VI. CONCLUSION

The use of PKC within the SNMP framework is unspecified under the current SNMPv3 standards. This paper outlines the features in PSM, a proposed method of incorporating PKC into the SNMP framework. Per-session authentication and encryption keys as well as stronger cryptographic algorithms in PSM strengthen the security model for SNMP transactions. Consequently, PSM provides greater security with a minimal increase in overhead compared with existing approaches, while maintaining the essential semantics of the SNMPv3 User Security Model.

VII. REFERENCES

[1] W. Stallings, "SNMPv3: A security enhancement for SNMP," *IEEE Communications Surveys*, Vol. 1 No. 1, Fourth Quarter 1998, http://www.comsoc.org/pubs/surveys/4q98issue/pdf/Stallings.pdf

[2] D. Harrington, R. Presuhn, B. Wijnen, An Architecture for Describing SNMP Management Frameworks, RFC 2571, *The Internet Society*, Apr 1999, http://www.ietf.org/rfc/rfc2571.txt

[3] U. Blumenthal, B. Wijnen, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), RFC 2574, *The Internet Society*, Apr 1999, http://www.ietf.org/rfc/rfc2574.txt

[4] B. Wijnen, R. Presuhn, K. McCloghrie, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), RFC 2575, *The Internet Society*, Apr 1999, http://www.ietf.org/rfc/rfc2575.txt

[5] E. Davis, "SNMPv3 - User Security Model," *Sys Admin – The Journal for UNIX System Administrators*, May 2000, http://electricrain.com/edavis/usm.html

[6] E. Davis, "SNMPv3 – View Access Control Model," *Sys Admin – The Journal for UNIX System Administrators*, June 2000, http://electricrain.com/edavis/vacm.html

[7] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 2nd Ed., Upper Saddle, NJ, USA: Prentice Hall, 1998.

[8] IEEE P1363: Standard Specifications For Public Key Cryptography, IEEE P1363 Working Group, http://www.manta.ieee.org/groups/1363

[9] W. Dai, Crypto++ 3.2 (C++ Cryptography Toolkit), http://www.eskimo.com/~weidai/cryptlib.html

[10] A. Goh, W. K. Yip, "Security Mechanisms for Telemedicine and Medical Informatics over Insecure Public-Access Networks," *Proc. Int'l Medical Instrumentation and Imaging Technology Exhibition and Conference (IMIIT)*, Kuala Lumpur, Malaysia, Aug. 1999.