MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Energy/Security Scalable Mobile Cryptosystem

Qiang Huang, Hisashi Kobayashi, Bede Liu, Daqing Gu, and Jinyun Zhang

TR-2003-79 February 2004

Abstract

Time-sensitive mobile commerce is vulnerable to message authentication delays. Significant power consumption incurred by cryptography is another limiting factor of most mobile devices. In this paper, we present a scalable mobile cryptosystem, which installs a group key and an elliptic curve private/public key pair in each device to enable both symmetric key and public key cryptography. We propose scalable key establishment protocols and secure routing protocols with scalable authentication schemes to make tradeoffs between security and energy, according to different mobile applications.

Publication History:

1. First printing, TR-2003-79, February 2004

Energy/Security Scalable Mobile Cryptosystem

Qiang Huang, Hisashi Kobayashi and Bede Liu
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544, USA

Daqing Gu and Jinyun Zhang
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139, USA

***Abstract*—Time-sensitive mobile commerce is vulnerable to message authentication delays. Significant power consumption incurred by cryptography is another limiting factor of most mobile devices. In this paper, we present a scalable mobile cryptosystem, which installs a group key and an elliptic curve private/public key pair in each device to enable both symmetric key and public key cryptography. We propose scalable key establishment protocols and secure routing protocols with scalable authentication schemes to make tradeoffs between security and energy, according to different mobile applications.**

I. INTRODUCTION

Secure and fast transmission of sensitive digital information over wireless channels has become increasingly important. The use of public key cryptography consumes a significant portion of the overall system resource. The computation complexity of symmetric key based operations is negligible, but the key management for symmetric key based system is complicated, and is always subject to attacks by adversaries.

A practical mobile cryptosystem should provide the customer flexible choices for balances between security, performance and power efficiency. The security requirements are different for different information to transmit, under different circumstances, or with different available resources. For instance, a customer may wish to use a pre-installed group key to authenticate his message when the battery is low, or the message is not important, while risk the security compromise of a node inside the group. It should also be able to operate several different tasks for different security and energy tradeoffs. For applications with a low security requirement such as toys/games automation and small home light control network, energy efficiency is more important and we can use symmetric cipher encryption and authentication. In security sensitive deployments such as military service, oil site operation, and hospital monitoring, it must be able to do asymmetric cipher operation for a stronger and scalable security feature. Therefore, in this paper we present a scalable mobile cryptosystem, which installs a group key and an elliptic curve private/public key pair into each device before they enter the mobile network, to enable both symmetric key and public key cryptography.

In section 2 of this paper, we propose scalable key establishment protocols in mobile network for applications with different security and energy requirements. Section 3 presents secure routing protocols with scalable authentication schemes. A group key is used to authenticate data with less security requirements, or when the node is in low power status. For critical commercial and military applications we propose a secure routing protocol with faulty link detection based on an efficient authentication scheme combing the use of elliptic curve cryptography (ECC) and hash functions. Section 4 summarizes this paper.

This work is supported by the New Jersey Center for Wireless and Internet Security (NJWINS) and Mitsubishi Electric Research Laboratories.

II. SCALABLE KEY ESTABLISHMENT

In this section, we propose three different key establishment protocols for mobile networks according to different application scenarios, the first one based on pure symmetric key cryptography and suitable for toys/games home networking, the second based on a hybrid of symmetric key and public key cryptography and designed for residential or small commercial wireless network deployment, and the last one based on pure public key cryptographic operations and targeting large industrial or military applications.

Perrig et al. present to use trusted third parties to assist node-to-node key agreement [1]. We call this trusted third party a security manager. A security manager is granted special capabilities to assist in provisioning link keys to other end mobile devices on-site. The security manager should first establish a link key with an end device before it can install link keys into that device for secure communicating with other end devices inside the mobile cluster. One way to accomplish the initial link key establishment task is to pre-install a master key table into each device. However, mobile networks may be highly versatile, involving temporary communications between devices that may have never met before. Thus we cannot predict and install all master keys needed for devices before they join the network, especially for large-scale wireless ad hoc networks. An alternative way is to use a shared group key [2] that is pre-loaded into each device in our proposed cryptosystem. Then when two devices want to establish a link key, they use this group key to encrypt and exchange their ephemeral key contribution data. Since the group key is fixed, the trust relationship is based merely on the knowledge of the other device's extended IEEE 64-bit address. The computation complexity and power consumption of symmetric key cryptographic operations are negligible when compared with public key schemes. However, a common group key poses a security risk if any one device is compromised. Therefore, this pure symmetric key based key establishment protocol in applications that require the least security protection, such as the home toys/games automation.

The use of asymmetric keys along with digital certificates to establish individual link keys can help reduce this risk and restrict the impact of key compromise to the compromised node itself, rather than to all its key-sharing parties. Public-key operations are quite expensive though. In recent years, ECC based key agreement protocols have gained popularity in constrained mobile environments, due to the property of small key sizes. In [3], we proposed a hybrid key establishment protocol, which is based on a combination of ECC and symmetric key operations. The motivation is to exploit the difference in capabilities between security managers and end devices, and put the cryptographic burden where the resources are less constrained. End mobile devices are much more battery and computational resources limited. However, the security manager means powered and more computational powerful. The hybrid key establishment protocol reduces the high cost elliptic curve random point scalar multiplications at the end device side and replaces them with low cost and efficient symmetric key based operations.

To prevent the impersonation attack, we use certificates in our key-establishment protocol, which provide a mechanism to check cryptographically to whom the public key belongs and if the device is a legitimate member of a particular network. In our mobile cryptosystem, we use the elliptic curve implicit certificate scheme [4], because of the resulting low communication complexity, which is a dominant factor for low bit transmission channels in sensor networks.

First, an elliptic curve E defined over $GF(p)$ (where p is the characteristic of the base field) with suitable coefficients and a base point P of large order n is selected and made public to all users. CA selects a random integer q_{CA} as its static private key, and computes the static public key $Q_{CA} = q_{CA} \times P$. To obtain a certificate and the static private-public key pair, an end device U randomly selects a temporary key pair (g_U, G_U) and sends G_U to CA. CA verifies U's identity and the authenticity of the request received from U. CA also selects a temporary key pair (g_{CA}, G_{CA}) and computes the elliptic curve point $B_U = G_U + G_{CA}$. The implicit certificate IC_U for U is constructed as the concatenation of CA's static public key Q_{CA}, the device identity ID_U, the elliptic curve point B_U and the certification expiration date t_U, i.e., $IC_U = (Q_{CA}, ID_U, B_U, t_U)$. CA then applies a one-way hash function H on IC_U and derives an integer $e_U \in [2, n-2]$ from $H(IC_U)$ following the conversion routine described in Section 4.1.3 of [5]. Finally, CA computes U's private-key reconstruction data $s_U = g_{CA} e_U + q_{CA} \pmod n$, U's public key $Q_U = e_U B_U + Q_{CA}$, and sends s_U and IC_U back to U. After U obtains the implicit certificate from CA, it computes the hash value $H(IC_U)$ and derives an integer e_U from $H(IC_U)$ following the conversion routine described in Section 4.1.3 of [5]. U also computes its static private key $q_U = s_U + g_U \cdot e_U \pmod n$ and its public key $Q_U = q_U \times P$. U then reconstructs the public key $\hat{Q}_U = e_U B_U + Q_{CA}$. If $\hat{Q}_U = Q_U$, U accepts the certificate and outputs the static key pair (q_U, Q_U); otherwise it rejects the certificate. By repeating the very same process, a security manager V acquires its certificate IC_V and static key pair (q_V, Q_V).

The certificate generation processes for end device U and security manager V are performed offline and before they join the network. When they first communicate to each other, they execute our hybrid key establishment protocol as below:

1. U and V send to each other their implicit certificates. The content of the certificate is verified at the other side, including the device identity and the validity period. If any check fails, the protocol is terminated.

2. V computes the hash value $H(IC_U)$ and derives an integer e_U from $H(IC_U)$ following the conversion routine described in Section 4.1.3 of [5]. V then obtains U's public key $Q_U = e_U B_U + Q_{CA}$. After performing the certificate processing, V can conclude that Q_U is genuine, provided that U later evidences knowledge of the corresponding private key q_U.

3. U selects a k-bit random number c_U as its link key contribution and a random $160-k$ bit integer r. U calculates its ephemeral private key $d_U = H(c_U \| r)$ and ephemeral public key $D_U = d_U \times P$, where H is a cryptographic hash function to map a binary string to a random integer $\in [2, n-2]$. U verifies V's certificate and obtains V's public key the same way as V does, but instead of computing Q_V directly, U computes $R = d_U \times Q_V = (d_U e) \times B_V + d_U \times Q_{CA}$. U can conclude that R is calculated from genuine Q_V, provided that V later evidences knowledge of the corresponding private key q_V. U then encrypts c_U by using the provably secure elliptic cure encryption [6], and sends to V $E = (D_U, (c_U \| r) \oplus R.x) = (E_1, e_2)$.

4. V decrypts the received message and obtains R by calculating $q_V \times E_1 = q_V d_U \times P = d_U \times Q_V = R$. V then computes $u = e_2 \oplus R.x$, and checks if $E_1 = H(u) \times P$. It yes, V gets c_U as the most significant k bits of u. Otherwise, the protocol is terminated. V then selects a k-bit random number c_V as its link key contribution, and encrypts c_V concatenated with its identity ID_V using symmetric key encryption under key c_U, generating $y = E_{c_U}(ID_V \| c_V)$. Note that proper encryption mode needs to be used, such as the Cipher Block Chaining (CBC) mode, which ensures that there is no way for any device W to derive $E_{c_U}(c_V)$ from $E_{c_U}(ID_V \| c_V)$ and change this value. V sends y to U.

5. V computes $MacKey \| LinkKey = KDF(c_U \| c_V \| ID_U \| ID_V)$, where KDF is the specified key derivation function, $LinkKey$ is the established link key, and $MacKey$ is for explicit key confirmation use. V then destroys c_U and c_V from its memory.

6. *U* decrypts the incoming message under c_U and checks if the decrypted message contains a proper coding of ID_V concatenated with some number. If the check fails, *U* terminates the protocol. Otherwise, *U* denotes the number as c_V, and *U* has verified that *V* has the knowledge of the private key q_V associated with Q_V. *U* computes $MacKey \| LinkKey = KDF(c_U \| c_V \| ID_U \| ID_V)$, and $z = q_U H(MacKey) + d_U \pmod n$. *U* then sends *z* to *V* and destroys c_U and c_V from its memory.

7. *V* verifies if $z \times P = h(MacKey) \times Q_U + E_1$. If it is false, *V* terminates the protocol. Otherwise, *V* believes that *U* has the knowledge of the private key q_U associated with Q_U, and *U* has provided the explicit key confirmation to *V*. *V* sends $z' = MAC_{MacKey}(ID_V \| ID_U)$ to *U*, where *MAC* is a message authentication code function.

8. *U* checks if z' is valid. If yes, *V* provides the explicit key confirmation to *U* and both sides take *LinkKey* as the final established link key and accept the connection.

In this protocol, authentication is accomplished by sending the challenge pairs (*E*, *y*) and (*y*, *z*). It is infeasible for an adversary to compute the correct response *y* without knowing q_V. Thus *U* can be sure that only *V* can produce the response and *U* verifies that *V* has the knowledge of the private key q_V associated with the certified Q_V. Also, $z \times P = H(MacKey) \times Q_U + E_1$ can be satisfied only if *z* is calculated by the correct private key d_U associated with the certified public key Q_U. Therefore, *V* can be sure that only *U* can produce the correct response. In addition, an adversary cannot obtain any information of c_U and c_V if both the symmetric and ECC encryption schemes are secure, which implies the link key contribution of each side is transferred securely to the other part.

This hybrid key establishment protocol consumes more node energy as compared to the pure symmetric key based protocol. However, since we verify the binding of the sensor's private key q_U to its public key Q_U in step 6 and 7 through a linear combination of the static key and the ephemeral key, rather than a multiplicative combination as in other ECC based pure public key protocols, at least one expensive elliptic-curve scalar multiplication of a random point is moved to the security manager side, and is replaced by one low cost modular multiplication, one modular addition and one symmetric key decryption. Therefore, our hybrid key establishment protocol is faster and saves more node energy than other public key based protocols, as evidenced by running our protocol on Mitsubishi's 16-bit single-chip microprocessor M16C. The whole protocol execution time on end device side is about 760 msec, while ECMQV protocol with ECC X509 certificates [4] and implicit certificates [5] takes 1110 msec and 1155 msec respectively, and the Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE) protocol [5] takes 1350 msec.

The hybrid key establishment protocol has much better security enhancement than our first pure symmetric key based protocol, while has moderate energy consumption on end mobile devices. We notice that if the security manager's private key is compromised, then all the link keys from earlier runs can be recovered from the transcripts. However, the corruption of the sensor node does not help to reveal the link keys. Therefore, our scheme provides half forward secrecy and is suitable to use in residential and small commercial mobile applications where security is important but not critical, and we can trade security for mobile users' energy efficiency.

To provide full forward secrecy, rather than being encrypted under a symmetric key c_U, c_V should be sent to *U* in a similar way that c_U is sent to *V* (i.e., through secure elliptic cure encryption [6]), and only *U* with its ephemeral private key can reconstruct it. Then our hybrid protocol is modified into a pure ECC based public-key key establishment protocol. However, this requires additional expensive elliptic curve random point multiplications on mobile user side, and is opposite to our purpose of offloading the computation burden of end devices. The pure ECC based public-key key establishment protocol is suitable to vital or security-sensitive network deployments, including natural disaster control, battlefield service, rescue missions, etc., where security is more important than energy efficiency.

III. SECURE ROUTING WITH SCALABLE AUTHENTICATION

Most of the proposed secure routing protocols in wireless networks are based on authentication in the route discovery process. Seldom work has been done to detect faulty links based on observation of misbehavior in the data forwarding phase. Awerbuch et al. [7] address the Byzantine failure problem by using adaptive probing techniques. Unfortunately, malicious nodes can differentiate probing packets and normal data packets and therefore can selectively forward the probing packets to avoid detection. Herzberg and Kutten [8] have proposed the combination use of acknowledgements, timeouts and fault announcements, to detect packet forwarding faults. The protocols are only presented in an abstract model, a realization of which is proposed by Avramopoulos et al. [9]. They propose a source routing protocol with Byzantine robustness by utilizing reserved buffer, sequence number and authentication of data and control packets based on message authentication codes. However, the authentication of data and control packets is based on message authentication code, which requires a separate authentication tag for each of the intermediate router, thus adding a lot of communication overhead when multi-hops are used.

To detect faulty links, we use acknowledgements, timeouts and fault announcements described in [8, 9]. However to reduce the communication overhead, we can still use a shared group key to authenticate all the data and control packets. With a fixed group key, the sender just needs to calculate and attach one authentication tag for each data and control packet. While this group key approach is efficient both in terms of

computation and communication overhead, it just mitigates outside attacks and does not protect against compromise of a single node. Therefore, it is more appropriate to use in applications with less security requirements, such as home automation applications, or when the node is in low power status and hence energy efficiency and performance are more important than security.

For commercial and military applications with a high security requirement, keeping the network available for its intended use is essential. We propose to use the Guy Fawkes protocol [10] for authentication, such that only a single authentication tag is attached for each data or control packet, and therefore can save more communication overhead than the detection scheme proposed by Avramopoulos et al [9].

We assume the dynamic source routing (DSR) protocol is used in the mobile network we envision. Now assume that the source *S* has a sequence of packets $\{m_1, m_2, \cdots m_n\}$ to send to the destination *D* through a source route $(S, n_1, n_2 \cdots D)$, where $n_1, n_2, \cdots$ are intermediate routers. When the source sends the first packet m_1, it selects two random passwords X_1 and X_2, sets a timeout to receive either a destination acknowledgement (ACK) or a fault announcement (FA) from a downstream router for this packet, and forwards to the first router the following message:

$$MSG_1 = \{m_1, h(X_1), Sig(m_1, h(X_1)), h(m_2, h(X_2), X_1)\}.$$

$Sig(m_1, h(X_1))$ is a signature over $(m_1, h(X_1))$ signed by the sender's private key. With its public key, every downstream router can verify that $(m_1, h(X_1))$ is valid and indeed generated by the claimed source node. Then each downstream router creates a new route table entry (S, e_1, e_2) associated with this source *S*, where $e_1 = h(X_1)$ and $e_2 = h(m_2, h(X_2), X_1)$, which will be used to authenticate the future message from the same source. An elliptic curve digital signature algorithm (ECDSA) can be used here due to a small size of the ECC key, faster processing speed and smaller communication complexity.

When sending the second packet m_2, the source selects another password X_3 and forwards the second message MSG_2 to the first downstream router:

$$MSG_2 = \{m_2, h(X_2), X_1, h(m_3, h(X_3), X_2)\}.$$

Each downstream router can verify $(m_2, h(X_2))$ by first applying the hash function *h* to X_1 received in MSG_2 and checking if the result $h(X_1)$ is the same value as e_1 stored in its route table. If yes, X_1 is valid. It then performs the hash function on $(m_2, h(X_2), X_1)$ received in MSG_2, and checks if the result is equivalent to the previously received value e_2 in MSG_1. If the check succeeds, the authenticity of $(m_2, h(X_2))$ is verified. Hence, the intermediate router knows that this message is indeed from the source node, and the content is not modified. It then updates the stored value of e_1 and e_2 as $e_1 = h(X_2)$ and $e_2 = h(m_3, h(X_3), X_2)$. The message MSG_2 is forwarded to the next hop as specified in the packet header.

Similarly, when sending the *k*-th ($k \geq 2$) packet m_k, the source selects a new password X_{k+1} and forwards the *k*-th message MSG_k:

$$MSG_k = \{m_k, h(X_k), X_{k-1}, h(m_{k+1}, h(X_{k+1}), X_k)\}.$$

When an intermediate router receives a data packet m_k, it verifies the authenticity of the packet by checking if $h(X_{k-1}) = e_1$ and $h(m_k, h(X_k), X_{k-1}) = e_2$. If both checks succeed, it updates its routing entry as $e_1 = h(X_k)$ and $e_2 = h(m_{k+1}, h(X_{k+1}), X_k)$. The packet is scheduled for transmission in the appropriate forward path. When the packet is transmitted, the router sets a timeout to receive either an ACK or an FA for this packet. ACKs provide feedback on whether a packet was successfully delivered. Timeouts detect delivery failures, which are set as the worst-case round trip time to the destination. With source routing the worst-case round trip time to the destination is known to the source and every intermediate router.

If any of the above checks fails, the packet is dropped. If the check at node n_i fails, there may be two reasons. The first is n_{i-1} modified $h(m_k, h(X_k), X_{k-1})$ in MSG_{k-1}, and the second is n_{i-1} modified $(m_k, h(X_k), X_{k-1})$ in MSG_k. In either case, the node n_i will drop the packet. Consequently, node n_{i-1} cannot get a valid ACK after timeout, and it will either report a link error of (n_{i-1}, n_i) by itself, or the node n_{i-2} will report an error of (n_{i-2}, n_{i-1}) to the source node. In either case, the detected fault link includes the malicious node n_{i-1}.

When the destination receives a data packet m_k, it verifies the authenticity of the packet in the same way as the intermediate routers do. If any of the checks fails, then the packet is dropped. If both checks succeed, it schedules an ACK for transmission along the reverse of the path that the packet traversed. The ACK reflects the packet identification number *k*. The destination also appends an authentication tag to the ACK whose purpose is to authenticate it to all upstream routers. The authentication tag bears the same structure as the one generated by the source. Specifically, when sending ACK_1 for the first packet m_1, the destination randomly selects two passwords Y_1 and Y_2, and sends the following information:

$$ACK_1, h(Y_1), Sig(ACK_1, h(Y_1)), h(ACK_2, h(Y_2), Y_1).$$

Similarly, $Sig(ACK_1, h(Y_1))$ is used to verify $(ACK_1, h(Y_1))$ to each upstream router. When sending acknowledgement for packet m_k ($k \geq 2$), the destination selects a new password Y_{k+1} and forwards:

$$ACK_k, h(Y_k), Y_{k-1}, h(ACK_{k+1}, h(Y_{k+1}), Y_k).$$

If the timeout at an intermediate router expires, it schedules for transmission to the upstream path an FA for the first downstream link. The FA reflects the identification number of the packet and also bears a similar authentication tag, for authentication the FA to upstream routers.

The other part of the protocol is the same as in [9]. When an intermediate router receives an ACK, it verifies its authenticity and that a timeout is pending for the corresponding data packet. If any check fails, it drops the ACK. Otherwise it cancels the timeout and further forwards the ACK. When an intermediate router receives an FA, it verifies its authenticity, it verifies that a timeout is pending for the corresponding data packet and that the link reported in the FA is the first downstream to the node that generated it. If any check fails, it drops the FA. Otherwise, it cancels the timeout and further forwards the FA.

If the timeout at the source expires, then it deletes the first downstream link from its Route Cache. It then finds a new path to the destination in its cached routes and reprocesses the "failed" packet as if it were a new packet. If the source receives an ACK_k, it assumes successful delivery of the packet m_k. If the source receives an authentic FA, then it deletes the link in the FA from its Route Cache, provided that this is the downstream link of the router that generated the FA. It then rediscovers a new path to the destination and reprocesses the "failed" packet.

Note that a prerequisite of our protocol is that MSG_{k-1} should always be received at each intermediate node before the source sending MSG_k, which is guaranteed if the source holds on transmitting MSG_k until it receives the ACK of FA regarding MSG_{k-1}.

We assume that each link has one a-priori reserved buffer for every source router in the network as also described in [9]. This ensures that normal packets are never dropped because of congestion. Authentication ensures that the reserved buffer is allocated to its intended source and protect against vicious flooding the network with unauthenticated packets. Malicious nodes that send packets frequently will soon use up all the buffer space allocated to them and the not served old packets will be discarded.

With authentication, the link containing a black hole or any passive attacker failing to forward packets to the destination can be detected since a malicious node does not hold the destination's secret key or password to be used, and thus cannot fabricate an ACK with a valid authentication tag.

The Guy Fawkes authentication tag also safeguards against replay. In a replay attack, an intermediate router stores authentic packets and introduces them at a later time into the network in order to "take out" new packets. In our protocol, a new packet is sent with a different password and the check on the replayed password fails when an intermediate node compares the hash of the password with the hash value it received in the previous message.

In our scheme, the authentication tag of each packet bears only two hashes and one password, while in the detection protocol introduced in [9], L authentication tags must be attached for L hops, and therefore, our scheme has a much smaller communication overhead. In the first step of our protocol, the authentication is based on ECDSA digital signature, while in later steps all authentications are done by symmetric key operations. Therefore, this scheme has a moderate computation overhead but with more security enhancement than our first routing protocol based on group key authentication.

IV. CONCLUSION

Scalable features are especially desirable for applications in low-power mobile cryptosystem. In this paper, we present a scalable mobile cryptosystem, which installs a group key and an elliptic curve private/public key pair into each device to enable scalable security processing. We propose scalable key establishment protocols and secure routing protocols with scalable authentication schemes, in which different security and energy tradeoffs are enabled for different application scenarios. The system user should choose the best appropriate protocol, by taking into account the level of security range required and the operational cost that the user is willing to accept.

REFERENCES

[1] A. Perrig, R. Szewczyk, V. Wen, D. Culler and D. Tygar. SPINS: Security protocols for sensor networks. Wireless Networks Journal (2002).

[2] S. Basagni, K. Herrin, E. Rosti and D. Bruschi. Secure pebblenets. Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001) (2001), 156-163.

[3] Q. Huang, J. Cukier, H. Kobayashi, B. Liu and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor netowrks. Proceedings of second ACM International Workshop on Wireless Sensor Networks and Applications (2003).

[4] Rene Struik and Gregg Rasor. Mandatory ECC Security Algorithm Suite, submissions to IEEE P802.15 Wireless Personal Area Networks, (March 2002).

[5] Certicom Research, Standard for efficient cryptography, SEC 1: Elliptic Curve Cryptography. Version 1.0, September 20, 2000.

[6] E. Fujisaki, T. Kobayashi, H. Morita, H. Oguro, T. Okamoto, S. Okazaki, and D. Pointcheval. PSEC: Provably secure elliptic curve encryption scheme. Primitive submitted to NESSIE by NTT Corp., (September 2000).

[7] B. Awerbuch, D. Holmer, C. Nita-Rotaru, H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. Proceedings of the 2002 ACM Workshop on Wireless Security (Sept. 2002).

[8] A. Herzberg and S. Kutten. Early detection of message forwarding faults. SIAM J. Comput., vol. 30, no. 4, (2000):1169-1196.

[9] I. C. Avramopoulos, H. Kobayashi, and R. Y. Wang. A routing protocol with byzantine robustness. The 2003 IEEE Sarnoff Symposium (2003).

[10] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. ACMOSR: ACM Operating Systems Review, vol. 32 (1998).