A secure communication protocol for ad-hoc wireless sensor networks

Pearce, Craig; Ma, Yin-Man; Bertok, Peter

Pearce, Ma, Y.-M., & Bertok, P. (2004). A secure communication protocol for ad-hoc wireless sensor networks. Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 79–84. https://doi.org/10.1109/ISSNIP.2004.1417441

Published Version: https://doi.org/10.1109/ISSNIP.2004.1417441

A Secure Communication Protocol for Ad-Hoc Wireless Sensor Networks

Craig Pearce, Vincent Yin-Man Ma, Peter Bertok

School of Computer Science and Information Technology
RMIT University
Melbourne, Australia
{crpearce,yma,pbertok}@cs.rmit.edu.au

Abstract

Security in wireless sensor networks is required for trust in collected data, however, resource limitations have made the security need of secondary importance. This paper proposes a new application-layer SPKI/SDSI protocol that provides secure communications, authentication and fast re-authentication. The protocol has been formally proven secure and results indicate the protocol to be suitable for wireless sensor networks.

Keywords: Wireless Sensor Networks, Communication Protocols, Ad-hoc Network Security

1. Introduction

Wireless sensor networks (WSNs) are typically embedded devices used for sensing and monitoring within environmental, health-care, military and physical science areas amongst others. Sensors are self-maintained and typically report collected data but do not accept network input. As sensors are placed in publicly accessible locations they need to be resistant to physical and network attacks. However, due to their size, sensors are limited by processing power, network connectivity, storage capacity and battery life. To preserve processing and communication speeds, security has been of secondary importance. This is a major concern as security is imperative for the trust of data collected and transmitted by network sensors.

We focus on providing a secure application-layer protocol for the communication between sensors on a WSN and a support network. Prior work shows that a dedicated proxy for each cluster of sensor devices allows for unification of communication between disparate devices [1]. Figure 1 shows the architecture we envisage, with a scenario of two wireless sensor network clusters that report to a process control service,

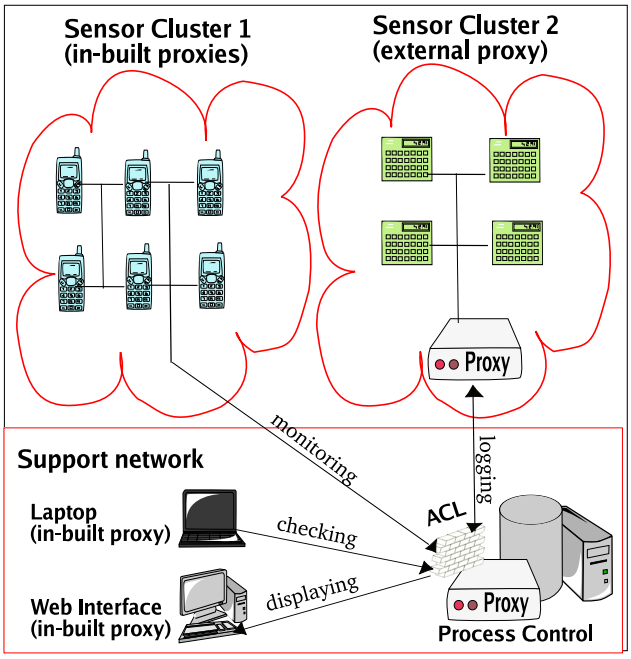

Fig. 1: Wireless sensor network (WSN) with external and in-built proxies

for example a chemical plant. It is crucial that data collection is trusted and reliable in such environments. One cluster contains sensors with in-built tamper-resistant proxies, while the other cluster contains less-powerful sensors with a dedicated proxy representing them. Network authorisation is achieved through an access control list (ACL) to ensure sensors, proxies and members of the support network are authorised to perform certain actions (proof of rights). The ACL allows controlled access to devices providing a shared resource, for example, the process control service. This allows another proxy (on behalf of their device) to make a request to the resource. Access

is granted or denied depending on whether the requesting proxy possesses the required credentials to prove authorisation. Simple Public Key Infrastructure / Simple Distributed Systems Infrastructure (SPKI/SDSI) certificates can provide fine-grained access control for authorisation of access requests to services, and its simplicity makes it a platform of choice in sensor networks.

Simplicity, however, leads to several security limitations, including lack of confidentiality, mutual authorisation and mutual authentication The SPKI/SDSI team suggested that mutual authorisation could be provided by replicating client-side authorisation, but in reverse. Prior work suggested tunnelling SPKI/SDSI over a transport layer security protocol, such as Secure Sockets Layer (SSL), for authentication, confidentiality and protection against replay and middle-person attacks [1]. This has since been proven to be impractical for mobile devices or wireless sensors due to significant processing overheads, and SPKI-SECURE has been proposed to provide confidentiality and authentication to SPKI/SDSI [2].

Significant work is required to minimise performance overheads of SPKI-SECURE to be acceptable on WSNs. In this paper, we resolve latency concerns of a secured SPKI/SDSI protocol by creating a fast re-authentication protocol for subsequent resource requests, named SPKI-SECURE-FAST. A fully-scaled implementation and experiments show that SPKI-SECURE-FAST is suitable for wireless sensor networks, where latency due to resource constraints were a limiting factor in the past.

Subsequently, we expect the adoption of SPKI-SECURE-FAST on wireless sensor networks to bring much needed trust of sensor data collection with optimised performance.

2. BACKGROUND

While we research application-layer network security, acknowledgement is made to recent security work at lower layers. Routing [3], key establishment [4], [5], network-level confidentiality and data authentication [6] and certificate chain discovery [7] have shown promising results, however application-layer authentication and authorisation are still open issues. If desired, SPKI-SECURE-FAST can build on these lower-level protocols in a layered fashion.

SPKI/SDSI is a simplified public key infrastructure (PKI) designed for fine-grained access control in ad-hoc networks [1]. Instead of using a pre-defined global namespace, certificates are issued by users and given an identifying name that is local in context. Certificates can be linked between local name spaces, giving powerful, distributed certificate chains with global scope.

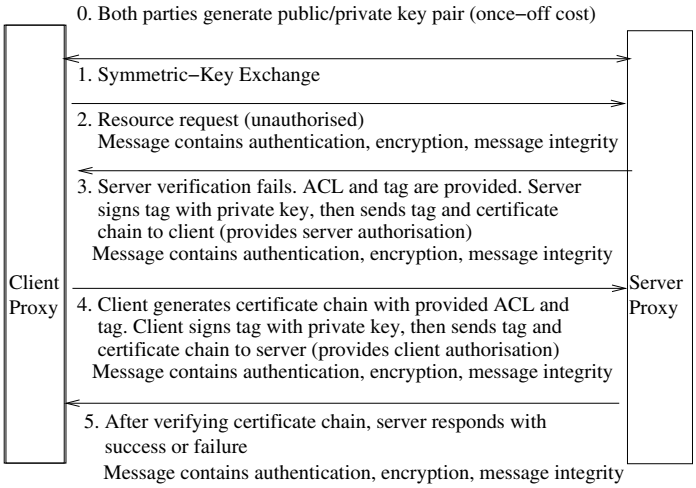

Fig. 2: Schematics of the Secured SPKI/SDSI Protocol [2]

Sensors are more interested in what they can currently monitor, as opposed to inter-communication, nonetheless, global identity is still a useful feature.

Figure 2 shows the secured SPKI/SDSI message exchange for two proxies communicating on behalf of their mobile devices, which builds on the original insecure protocol illustrated elsewhere [1]. Each message communicated for a resource request is encrypted and signed to provide confidentiality and authentication. Mutual authorisation is met by the (1) sensor-proxy (*SP*) sending their chains of name and authorisation certificates proving they are allowed to perform the request and (2) server sending their certificate chain providing they are allowed to authorise requests.

Due to the amount of public key cryptographic operations involved for each resource request, latency is a concern for wireless sensors. By optimising the protocol, we can make significant savings in processing and communication costs for subsequent resource requests between sensors and the support network.

3. PROPOSED PROTOCOL

To resolve latency concerns of security between the Transport and Applications Layers for SPKI/SDSI in ad-hoc wireless sensor networks (WSNs), we have modified SPKI-SECURE to support fast re-authentication of subsequent resource requests.

The protocol extension works by using SPKI-SECURE's key-exchange protocol as a source to generate a list of one-time secret keys for encryption of subsequent resource requests.

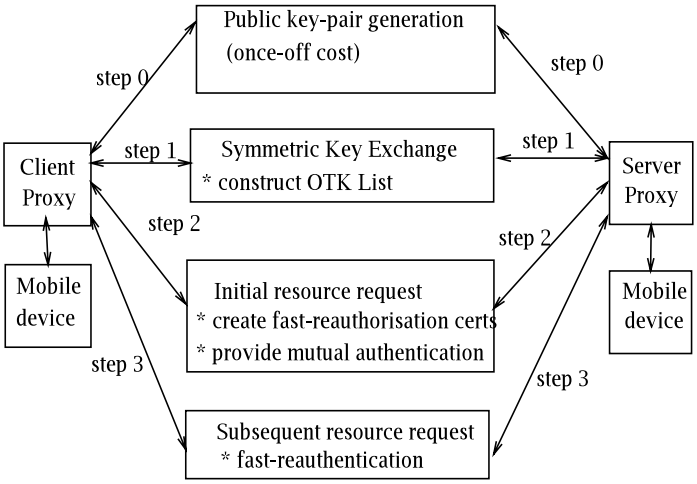

Fig. 3: Symmetric Key-Exchange Protocol

Figure 3 shows a diagram for the symmetric key exchange protocol.

For a sensor-proxy (*SP*) to request access to a resource governed by a server-proxy for the first time, the following steps will take place:

1) *SP* performs Service Discovery (not discussed in this paper) to locate the nearest server governing the resource requested by the *SP*;
2) If not already done, both *SP* and server generate their public/private key-pairs for public key cryptographic support (Figure 3 step 0);
3) Both parties perform a symmetric key exchange by sharing public keys (Figure 3 step 1). We use the Diffie-Hellman key agreement protocol as this allows for a shared secret to be established without having to communicate secrets over an insecure channel;
4) *SP* and server use the symmetric key as a master key and generate a list of one-time keys (OTK List) by repeatedly hashing the master key (described shortly). The session ID and *SP* ID is assigned by the server. Hashing the *SP's* public key will provide a unique *SP* identification value (assuming that both the *SP* public key is unique and a secure message digest function is used);
5) *SP* now provides a resource request to the server using the next available OTK from the OTK List as a secret key for message integrity and confidentiality (Figure 3 step 2). When *SP* and server certificate chains are validated, each issues the other with a re-authorisation certificate. This allows subsequent resource requests to validate a single re-authorisation certificate instead of a chain of certificates, which in turn reduces processing overhead. This suggestion was made by the SPKI/SDSI team for a server validating a *SP*. We have extended the notion to operate for validating both server to *SP* and *SP* to server;
6) Subsequent resource requests (Figure 3 step 3) involve a request encrypted with the next OTK in the OTK List. Performance is significantly improved as authentication, exchange of public keys and secret key generation has already taken place and there is no need for public key cryptographic operations;

Fast re-authentication is implemented as a more secure approach to SSL's session resumption. Fast re-authentication is enabled by:

1) both *SP* and server generating a OTK List in the symmetric key exchange
2) checking authenticity and authorisation in the initial resource request
3) using re-authorisation certificates and not repeatedly using public key operations for re-authenticating in subsequent requests
4) authenticating based on knowledge of the next OTK in the OTK list which is much faster than repeated use of public key cryptography

One-time keys are generated by both *SP* and server independently and not transmitted over the communication channel. However, if the *SP* did not have the resources for OTK list generation then the protocol would support the server generating the OTK List and transporting it to the *SP*. Similarly, the *SP* could easily generate the OTK List and communicate it to the server if the server was burdened with too many concurrent requests. Both choices could be determined at runtime, however, here we assume the *SP* is powerful enough. Obviously the OTK List would need to be encrypted if it was communicated (by using the shared key generated by the symmetric-key exchange protocol).

Table 1 gives an example of OTK List generation with a formal description showing hashed output to be the input for the next generation and applied in an iterative fashion. The list is reversed once enough keys have been generated. Using a secure hash function provides the 'one-way' property whereby it is computationally easy to generate the hash output, but infeasible to determine the hash input given the hash output [8]. As a result, subsequent encrypted communication is not broken if an attacker obtains a prior key.

We also extend usage of fast re-authorisation, a concept already introduced to SPKI/SDSI [1]. It involves the server

TABLE 1: ONE-TIME KEY (OTK) LIST GENERATION AND USAGE

$$\overset{\textbf{Generated}}{\longrightarrow}$$

$$a_1 = f(a_0) \;\; a_2 = f(a_1) \;\ldots\; a_n = f(a_{n-1})$$

$$\overset{\textbf{Used}}{\longleftarrow}$$

proxy creating a new certificate for the *SP* once the *SP* has successfully been authorised. The certificate states the *SP* is authorised to perform the requested operation on the resource and is signed by the server proxy. This is then used within subsequent resource requests to save repeated chain generation and validation. Due to resource constraints, fast re-authorisation is crucial to reduce public key operations on mobile devices.

A. Applicability to wireless ad-hoc sensors

The proposed protocol is within the limitations of WSNs, as shown below:

- **Memory requirements:** Both *SP* and server proxy will require a state table that records session information, an OTK List and an identifier noting which OTK will be used for the currently negotiated or next resource request. Devices will need enough storage capacity to contain the Java Virtual Machine, which is the implementation platform used for the SPKI/SDSI framework, including SPKI-SECURE cryptographic functions.
- **Processing Costs:** The most expensive operations required involve public key cryptographic operations of key generation (a once-off cost), public key encryption and decryption. Public key operations have been minimised with:
  - one public and one private key operation needed for fast re-authorisation
  - one private and public key operation for authentication of initial resource requests
  - only secret key operations for subsequent resource requests

  Fast and computationally less intensive algorithms can be used with our protocol (for example, Extensible Tiny Encryption Algorithm), but there is always a trade-off between security and speed. Completely removing public key cryptography is not favoured as it provides certificate-based access control, painless symmetric key distribution and strong user authentication.
- **Network Connectivity:** In a wireless network, devices are connected to a mobile base station that provides network coverage to a specific region (known as a cell). Roaming devices disconnect and reconnect to the network as they traverse cell perimeters. We chose TCP as the transport protocol due to its reliability. Network disconnection involves connection re-establishment through the TCP 3-way handshake and a repeat of the message exchange for the current protocol invocation (be it symmetric-key exchange, initial resource request or subsequent resource request). Recent work in the viability of TCP/IP connectivity for wireless sensors has shown proxies to be a promising approach [9]. Our protocol extension involves quicker secure connection establishment and significantly faster support for subsequent resource requests.

B. Informal Security Analysis

Taking an informal look at the message flow, we attempt to note weaknesses:

- We assume host security to be intact. While not researched here, host security is important for sensor networks as they are often in unmanned, but publicly accessible environments. Tamper-proofing is a popular method of enabling host security;
- Care was taken to ensure it was as simple as possible. Complex protocols make security analysis increasingly difficult to perform, resulting in missed security flaws. Prior analysis showed SSL to be rather complex and burdensome, if not suitable for wireless sensors [2]. Reducing the number of exchanged messages, non-deterministic choices and public key operations helped make the protocol both simpler in design and computationally faster than SSL, as our results will show.
- Due to keys changing for each request, an attacker cannot hope to gain enough ciphertext over time to perform a ciphertext-only attack (gathering a large collection of ciphertext encrypted with the same key will have periodic patterns over time);
- If an attacker discovers the shared symmetric key from the symmetric key exchange and then catches the first resource request message, they can decrypt the message and have the entire list of one time passwords. This allows the attacker to view, and possibly modify, resource requests and responses. This problem is difficult to circumvent. One solution is to limit the number of fast re-authentications. Such attacks involve breaking host security, an act that our research assumes safe, or discovering an unknown weakness within the Diffie-Hellman key agreement protocol;
- Replay attacks involve an eavesdropper catching a message in transit and replaying it at a later time to attempt to gain unauthorised access. Using OTKs prevents this as a replayed message will be encrypted with an out-of-date

key. Other ways to prevent replay attacks include using timestamps (requires synchronised clocks and a network time server) and exchange of randomly generated nonces.
- A passive eavesdropper can perform traffic analysis and learn which parties are communicating with each other. An underlying privacy or security protocol, such as [10], could be used but adds noticeable performance penalties;
- We assume message digest functions (also known as one-way hash functions), symmetric key ciphers and public key ciphers are secure enough to avoid cryptanalysis in a reasonable amount of time. This assumption is plausible, as the cipher in question should not be used if found to be weak. Additionally, our protocol has been designed with modularity as a requirement. New ciphers can be "plugged in" as replacements where desired;
- We assume the pseudo-random number generator (PRNG) contains enough entropy to generate random-like sequences that are not predictable in a tractable amount of time;

4. Formal verification

SPKI-SECURE-FAST has been formally verified with the Casper and FDR2 toolset [11]. Casper is a tool that simplifies the description of the Communicating Sequential Processes (CSP) process algebra, a popular language to formally represent communications protocols. The Casper-represented CSP is then input into FDR2 which performs a brute-force state-space search for attacks against security assertions defined by the protocol designer.

Formal analysis proved that our assertions of confidentiality and authentication held.

5. Implementation

An entire SPKI/SDSI architecture has been written in Java, named Ad-hoc Trust Management System (Adtrust), which supports fine-grained access control. The implementation provides a choice of the four SPKI/SDSI protocols discussed in this paper:

1) original SPKI/SDSI protocol (with noted security limitations)
2) SPKI/SDSI tunnelled over SSL
3) SPKI-SECURE (researched elsewhere [2])
4) SPKI-SECURE-FAST (with newly researched fast re-authentication)

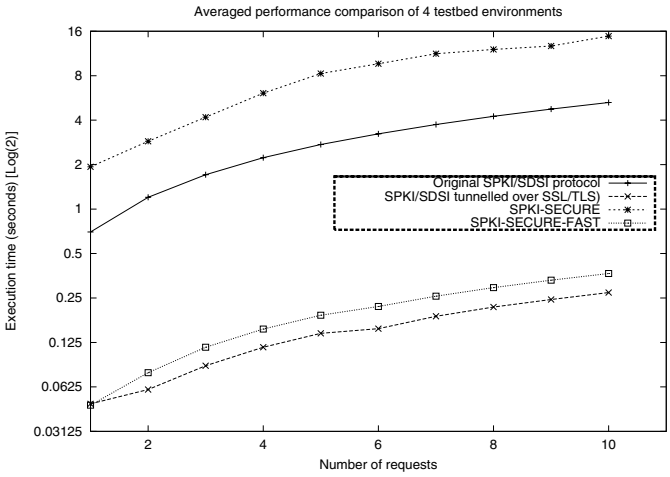

Fig. 4: 1Mbps wireless communication

Cryptographic support is provided by the Java 1.4.2 Cryptographic Architecture (JCA) and a third-party BouncyCastle security provider.

The Diffie-Hellman key exchange uses RSA (PKS#1) [12]. We plan to replace this with static-static Authenticated Diffie-Hellman to prevent middleperson attacks [13].

6. Experimental Results and Discussion

The purpose of our experiments is to compare times of processing and communication capabilities of mobile devices between the four SPKI/SDSI protocols. They will show how efficient our security extensions are in comparison to previously suggested ideas in securing the SPKI/SDSI message dialogue exchange.

To simplify data collection, experiments were simulated on Intel(R) PIII 1000MHz machines, presenting the scenario of wireless sensors using their trusted proxy for intensive operations. Communication between sensor and associated proxy is outlined elsewhere [1]. Network communication was conducted over a 802.11g wireless network with a speed of 1Mbps. The wireless AP used Media Access Control (MAC) address filtering and Wired Equivalency Privacy (WEP) based encryption for additional security.

Figure 4 shows a graph of relative performances of each protocol in regards to the total amount of time it took for a *SP* to send varying numbers of serialised resource requests. Measurements were collected on ten occasions, with a statistical average being taken to help cancel the effect of time variations due to underlying variables. SPKI-SECURE-FAST and SPKI-SSL

are the fastest as they minimise communication overheads of subsequent resource requests and provide fast re-authentication. SPKI-SECURE is the slowest due to additional cryptographic overhead. The three secured SPKI/SDSI protocols all provide encryption of the complete data exchange and a message digest, on top of the existing SPKI/SDSI's support of access control.

It is evident that SPKI-SECURE-FAST and SPKI-SSL become more efficient as the number of requests grow. This is a significant result as it shows SPKI/SDSI with additional security is actually faster than the original SPKI/SDSI protocol with missing network security functionality.

We assumed proxies to be embedded into their wireless sensors, which eliminated the need of porting the implementation to ARM processors and having an existing wireless sensor network.

In our experiments we used the AES-128 (Advanced Encryption Standard) symmetric-key algorithm with Cipher Block Chaining (CBC) mode, RSA (Rivest Shamir Adleman) public key algorithm and the SHA1 (Secure Hash Algorithm) message digest function. In resource constrained WSNs, these can be replaced by simpler components, such as TinyOS (including Java 2 Micro Edition), and the Extended Tiny Encryption Algorithm (XTEA) for encryption and hashing.

7. Conclusion

SPKI-SECURE-FAST improves the state of security within SPKI/SDSI without sacrificing performance, making it appealing for resource-limited wireless sensor networks. We succeeded in providing fast re-authentication for subsequent resource requests, significantly reducing overheads.

The formal and informal analyses proved that the proposed security extensions work well and do not have negative impact on the protocol operation.

Results indicated our solution was suitable for resource-constrained wireless sensors networks.

One Time Key (OTK) Lists enabled fast re-authentication and did not rely on expensive public key cryptography. Additionally, OTKs have the security feature of using keys only once for each resource request and greatly reduced the possibility of cryptographic attacks by informed enemies.

We anticipate results from our work to encourage further research into using the SPKI/SDSI access control framework within ad-hoc wireless sensor networks.

A. Further work

Investigating risks and countermeasures of denial of service attacks in SPKI-SECURE-FAST and other sensor network security protocols would increase reliability of WSNs.

Acknowledgements

Distributed Computing laboratory facilities were partially provided by Sun Microsystems AEG grant number 7832-030217-AUS.

Finally, we would like to thank Formal Systems for providing a license to freely use FDR2.

References

[1] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest. Proxy-based security protocols in networked mobile devices. In Proceedings SAC (Symposium on Applied Computing)., 2002.

[2] C. Pearce, P. Bertok, and C. Thevathayan. A Protocol for Secrecy and Authentication within Proxy-Based SPKI/SDSI Mobile Networks. AusCERT Asia Pacific Information Technology Security Conference ISBN: 1864997745, May 2004.

[3] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, May 2003.

[4] L. Eschenauer and V. Gligor. A Key Management Scheme for Distributed Sensor Networks. In ACM Conference on Computer and Communications Security. Washington D.C., November 2002.

[5] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. To appear in Proc. of the IEEE Security and Privacy Symposium, 2003.

[6] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[7] H. Huang and S. F. Wu. An approach to certificate path discovery in mobile ad hoc networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 41–52. ACM Press, 2003.

[8] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2003. ISBN 0-201-44099-7.

[9] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller. Connecting Wireless Sensornets with TCP/IP Networks. 2nd International Wired/Wireless Internet Communications Conference, February 2004.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In Proceedings of the 13th USENIX Security Symposium, August, 2004.

[11] B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using Casper and FDR. In Workshop on Formal Methods and Security Protocols, July 1999.

[12] RSA Security. RSA Cryptography Standard: PKCS 1 `http://ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`. Technical report, June 2002.

[13] R. Housley. RFC 3370: Cryptographic Message Syntax (CMS) Algorithms. The Internet Society. `http://www.ietf.org/rfc/rfc3370.txt`. Technical report, August 2002.