Secure End-to-End Transport Over SCTP

Carsten Hohendorf, Erwin P. Rathgeb
University of Duisburg-Essen
Institute for Experimental Mathematics
Computer Networking Technology Group
45326 Essen, Germany
Phone: +49-201-183-7637
Email: {hohend, erwin.rathgeb}@iem.uni-due.de
Esbold Unurkhaan
Mongolian Science and Technological University
Computer Science and Management School
P.Box 313/49 Ulaanbaatar, Mongolia
Email:esbold@csms.edu.mn
Michael Tüxen
Münster University of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt, Germany
Phone: +49-2551-962-550
Email:tuexen@fh-muenster.de

***Abstract*— The Stream Control Transmission Protocol is a new transport protocol initially developed to transport signaling messages over IP networks. The new features of SCTP make it also a suitable candidate for applications which nowadays use the standard transport protocols TCP and UDP. Many of these applications have strict requirements with respect to end-to-end security. Providing end-to-end security by using IPsec or the Transport Layer Security (TLS) protocol in combination with SCTP is subject to functional and performance related limitations. These can be avoided by integrating security functions directly into SCTP (S-SCTP). Although S-SCTP in principle solves all limitations, some issues remain hindering broad deployment of this solution. Therefore, we propose an alternative solution which preserves the advantages of S-SCTP while avoiding major modifications to existing standards and operating systems.**

***Index Terms*— End-to-End Security, SCTP, TLS, IPSec, DTLS.**

I. Introduction

The Stream Control Transmission Protocol (SCTP) is a new transport protocol, which has been approved by the IETF as a proposed Standard [5] in 2000. Originally SCTP was developed to transport telephone signaling messages over an IP network. The goal was to provide a similar reliability and quality of service like an SS7 signaling network. The original framework for the SCTP definition is described in [4].

Although SCTP has been developed to transport signaling messages, it is a general purpose transport protocol with distinctive features which make it suitable for many applications currently using the classical transport protocols TCP and UDP. The first protocol other than signaling transport (SIGTRAN) - to standardize the use of SCTP was Reliable Server Pooling (RSerPool, [7]). The use of SCTP is also defined for the Authentication, Authorization and Accounting (AAA) protocol [6] and the IP Flow Information Export (IPFIX) protocol [30]. Due to the fact that SCTP is already available in most of the major operating systems (Linux, FreeBSD, Solaris, Cisco IOS), it can be anticipated that other applications for SCTP will follow soon. Most of these applications have strict security requirements which are, e.g. for SIGTRAN [12] and RSerPool [24], already specified in standards documents. Therefore, it is crucial for the success of SCTP to provide an efficient and flexible security solution which supports all features of SCTP.

This paper shortly describes the already standardized SCTP security solutions, namely SCTP over IPsec [10] and TLS over SCTP [9] and identifies their limitations. It will be shown that these functional and performance related limitations can be overcome by integrating security functions directly into SCTP as proposed by us in earlier publications under the name S-SCTP (see [26], [31] and [32]). One problem remaining with S-SCTP is that a full scale introduction would require these extensions of SCTP to be included in future operating system kernels. Therefore, and based on the discussions in the IETF [26], we propose an alternative security solution for SCTP which is based on the use of the newly defined Datagram TLS protocol (see [37] and [18]) in combination with the chunk authentication extension of SCTP [23] currently under standardization in the IETF. We will describe the concept of this "SCTP aware DTLS" solution in detail to substantiate its feasibility. In addition, we will discuss some aspects of an implementation based on OpenSSL.

This is a revised and extended version of a paper presented at the International Conference on Emerging Trends in Information and Communication Security (ETRICS), Freiburg, Germany, June 2006. [39] With kind permission of Springer Science and business Media.

II. Introduction to SCTP

In this section we shortly review some new features of SCTP which are relevant to end-to-end security. A more detailed SCTP description can be found e.g. in [8], [33] and [35].

An SCTP connection called ”association” is established by using a 4-way handshake protected by a cookie mechanism which makes it less susceptible to blind denial-of-service attacks. SCTP packets consist of a common header followed by a sequence of data units called ”chunks”. The association is managed by using specific control chunks while user messages are transported in data chunks. Multiple chunks can be bundled into one SCTP packet, so that the resulting SCTP packet best uses the Path Maximum Transmission Unit (PMTU). SCTP is message oriented and provides a more flexible data delivery than current transport protocols. An important feature to achieve this flexible data delivery is the streaming function of SCTP. With this function several message streams can be multiplexed into one association. Only the messages within one stream are delivered in sequence, so a message lost in one stream will not affect the delivery of messages in another stream. This eliminates the head-of-line blocking known from TCP. It is optionally also possible to deliver data out of order within a stream.

Another distinctive core feature of SCTP is multi-homing, i.e. the ability for a single SCTP endpoint to support multiple IP addresses. So an SCTP endpoint can maintain several network paths to its peer. Only one path, the primary path, is used for normal data transmission. The other paths are only used in the case of transmission failures. The benefit of multi-homing is a better protection of the association against network failures.

In addition to these standard features of SCTP, two extensions have been proposed which also have to be dealt with when providing end-to-end security for SCTP. These extensions are the Partial Reliability SCTP [11] and the ADD-IP extension [20].

The Partial Reliability extension (PR-SCTP) describes a mechanism to stop an SCTP endpoint from retransmitting specific data chunks which have not been acknowledged. A control chunk, called FORWARD-TSN (Forward Transport Sequence Number), is sent to the receiving side indicating that all chunks with lower sequence numbers will not be retransmitted, and that the receiver does not have to wait for them any more. The decision when to stop retransmitting the data is taken locally at the sender and is application specific. Therefore, only the signaling procedure is fully specified in [11] but not all possible methods (policies) describing when to abandon a data chunk. One obvious policy is based on a limited lifetime of a user message. This means that a data chunk is not (re)transmitted once its time-to-live has expired. Another policy can be to limit the number of retransmissions. These policies are particularly useful to transport real time traffic, where out of date data is useless anyway. A third possible policy is based on send buffer limits and different priorities for the user messages. If the send buffer is full and a new high priority message arrives, a message with lower priority gets deleted. This policy is e.g. used for IPFIX as described in [30].

The ADD-IP extension allows to dynamically reconfigure IP addresses of an existing SCTP association. Therefore, it becomes possible that the available paths between the endpoints change during an association lifetime. ADD-IP also allows to change the primary path of an active association. This extension was proposed to support long-lived associations, where it is sometimes necessary to change some network connections. In addition to that, also a new application for SCTP was presented on the basis of this extension, called Mobile SCTP [21]. Although the development of the ADD-IP extension already started at the end of 2000, it has not yet become an RFC. The main reason for this is that a security mechanism was missing allowing to solve the additional security issues introduced by the ADD-IP option. Such a mechanism providing authentication for specific SCTP control chunks has now been developed and is described in [23]. Its usage for the ADD-IP extension is mandatory.

The standard API for using transport protocols is the socket API, described for example in [34]. Since SCTP provides more features than UDP or TCP it was necessary to extend the socket API to support all features of SCTP. These extensions are described in [28] and the sections covering SCTP in [34]. With this extended socket API it is possible to write applications using SCTP which compile and run on a variety of Unix operating systems.

The SCTP implementations which are included in Solaris 10, Linux with 2.6 kernels and in the FreeBSD 7 source tree all support the basic protocol defined in [5] and [19] with the extensions PR-SCTP defined in [11] and ADD-IP defined in [20]. They all provide the socket API and the source code is available under different licenses.

III. Existing security solutions

In this section we will present the three security solutions already proposed, namely SCTP over IPsec, TLS over SCTP and S-SCTP. All of these solutions can use the same cipher suites and Hash MAC (HMAC) algorithms, so there is no difference in the provided security, as far as the algorithms are concerned. All of them provide similar mechanisms for key exchange and security session management. Therefore, the major difference with respect to security is that TLS over SCTP – residing on top of SCTP – cannot protect SCTP control information. However, there exist several functional and performance related differences and issues which will be discussed in the remainder of this section.

A. SCTP over IPsec

SCTP is typically used in IP based networks. If secure transfer is required, SCTP can utilize the IP security protocol suite [13], [14], [15] for integrity, authentication and confidentiality. To establish IPsec Security Associations (SAs), a key negotiation such as IKE [16] may be used. The management and handling of IPsec security

associations is complex even when TCP is used. Since SCTP has some features, like multi-homing, which are not well supported by IPsec, the management and handling of the SAs is even more complicated. The use of SCTP with IPsec is defined in [10]. This RFC identifies the problems of SCTP over IPsec, i.e. the management of IPsec SAs in the case of multi-homing and the support of the ADD-IP extension of SCTP. The proposed solution to these problems is to use a list of IP addresses in the security policy database instead of single IP addresses. However, there are no implementations available to date which fully support this RFC.

If the Authentication Header (AH) or the Encapsulation Security Payload (ESP) is used to provide security services for SCTP frames, SCTP is treated as just another transport layer protocol on top of IP (such as TCP, UDP, etc.). Without the proposed modifications to IPsec introduced in [10], this solution requires the configuration of multiple IPsec Security Associations (SA) to support a multi-homed SCTP association. In the OSI model IPsec is one layer beneath SCTP, so it is not capable of differentiating between application data that must be secured and data that does not need to be secured. As a result, IPsec secures all data traffic resulting in an increased computational effort. Another disadvantage of this is that each SCTP packet is secured separately by IPsec. So in the case of long messages which must be fragmented by SCTP the overhead increases since two or more SCTP packets per message have to be secured.

B. TLS over SCTP

RFC3436 [9] describes the usage of the Transport Layer Security (TLS, [2]) protocol over SCTP. TLS is designed to operate on top of a byte-stream oriented transport protocol providing a reliable, in-sequence delivery. Thus, TLS is currently mainly used on top of the Transmission Control Protocol [1].

TLS over SCTP uses one TLS session per stream. This potentially leads to performance problems when the association needs many secured streams. Every message is secured by TLS before it is sent over SCTP. If the application sends many small messages, each message is secured separately. This results in an increased overhead compared to a solution securing a complete SCTP packet including several bundled messages. Since each TLS record depends on the state of the previous record, the unordered delivery service of SCTP is not supported. For the same reason, the PR-SCTP extension cannot be used. In the OSI model, TLS is located above the transport layer, so it cannot protect SCTP control chunks or the SCTP common header as they are added after TLS passes the data to SCTP.

An advantage of TLS over SCTP compared to SCTP over IPsec is that this solution can mix secured and unsecured traffic within one SCTP association efficiently. The TLS user can also take full advantage of the multi-homing feature and the proposed Add-IP extension of SCTP without modification of TLS.

C. Secure SCTP (S-SCTP)

The usage of SCTP together with standard security protocols (TLS or IPsec) leads to significant limitations and potential inefficiencies as discussed above. Neither TLS nor IPsec support all SCTP features and due to multi-streaming at the upper service access point and multi-homing at the lower service access point, non-integrated solutions are always potentially inefficient in some scenarios. Therefore, the security extension S-SCTP was proposed by us in some earlier work [26], [31], [32]. S-SCTP integrates crypto functions into SCTP itself in an efficient and user-friendly way. This extension is designed to avoid the drawbacks of the non-integrated solutions, whilst still providing full compatibility with the original SCTP protocol when no protection is being used.

The secure session of S-SCTP is initialized after the normal SCTP association is established. If one endpoint does not support the S-SCTP extension or the setup of the secure session fails, e.g. due to wrong certificates, the application can decide if it wants to use the unsecured association or if it shuts down the association.

The basic concept of the S-SCTP solution is that an association has only one Common secure session for all data streams in a multi-streaming case and for all addresses in a multi-homing scenario. In order to achieve this, the security mechanism is integrated between the upper functional block of SCTP which performs grouping of SCTP chunks to SCTP packets (bundling) and the lower functional block which performs the selection of network paths by choosing a destination address to send the SCTP packet as shown in Fig. 1.

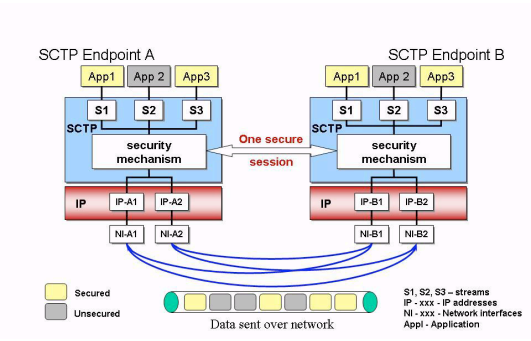

Figure 1. S-SCTP concept

S-SCTP provides the same security features as the two standardized security solutions, namely authentication, integrity and confidentiality. For that, S-SCTP uses the same standard cipher and HMAC algorithms as IPsec and TLS. To keep the protocol overhead of S-SCTP as small as possible it supports a flexible mix of secured and unsecured data, not only on a per-stream basis as TLS over SCTP, but even on a per chunk basis. To further reduce the overhead, chunks marked for encryption are grouped together and encrypted into one cipher text block. The HMAC is calculated per packet and not per chunk for the same reason. S-SCTP also offers a set of predefined

security levels, which are easy to select and, furthermore, can be changed during a secure session lifetime.

To avoid the complexity of secure session management known from IPsec S-SCTP provides the user with a simple API for configuration:

- simple initialisation, re-keying and termination of secure sessions,
- flexible choice of standard cipher suites,
- easy integration of newly defined cipher suites if required and
- simple selection and modification of security levels.

S-SCTP's only performance disadvantage compared to TLS over SCTP occurs when long messages have to be fragmented at the SCTP layer. In that case S-SCTP has to secure two or more packets separately, so the overhead is bigger compared to TLS where the message is first secured and then fragmented. With respect to all other criteria S-SCTP performs as good as or better than any of the other two security solutions.

The following table provides a summary of the qualitative comparison of the security solutions with respect to usability, overhead, management cost and performance. In the table "+" indicates that the feature is well supported by the solution, "-" denotes disadvantages of the solution with respect to the feature and "no" indicates that this feature is not supported at all. The "(-)" for the multihoming support of IPsec indicates that this problem is theoretically solved, however there are no implementations of this solution to date.

TABLE I.
COMPARISON OF SECURITY SOLUTIONS

Criteria	TLS	IPsec	S-SCTP
Scalability for multiple streams	-	+	+
Support for SCTP multihoming	+	(-)	+
Overhead for small messages	-	+	+
Overhead for long messages	+	-	-
Protection of unordered delivery service	no	+	+
Protection of SCTP control chunks	no	+	+
Flexible multiplexing of secure and insecure streams	+	no	+
Management of security sessions (handling, automation)	+	-	+
Partial Reliable Transport (SCTP extension)	no	+	+
Dynamic Address Reconfiguration (SCTP extension)	+	-	+

IV. QUANTITATIVE COMPARISON OF THE EXISTING SECURITY SOLUTIONS

In order to quantify the effect of the issues identified in Sect. III, a testbed has been set up and configured with the three security solutions. All three security solutions used the same crypto algorithm, namely the 3DES-SHA cipher. The testbed consisted of 2 Linux PCs (multi-homed) and a FreeBSD PC used as a router. The endpoint PCs had an Athlon AMD 2000 MHz processor and 512 MB of RAM, the router had a 64-bit AMD 3,0 GHz processor and 1 GB of RAM. All PCs were equipped with 1 Gbps Ethernet cards.

The tests were performed using a traffic generator sending random data to a traffic analyzer which calculated the throughput in 1 second intervals. Each point in the diagrams represents the average of a 5 minute measurement period which was repeated five times. The different link speeds used in the measurements were simulated by Dummynet [36] which was installed on the router. The

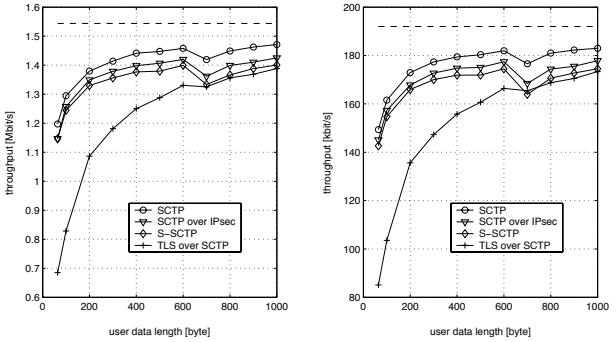

Figure 2. Throughput of security solutions over different links

measurements in Fig. 2 represent scenarios where the link is the bottleneck for the transmission which is typically the case for WAN connections. The left figure shows the throughput of the three security solutions over a T1-link with 1,544 Mbps and the right one over a DSL-uplink with 192 kbps (note the different scale on the y-axis). In such a scenario the throughput penalty for the three security solutions depends only on the overhead added to secure the data. As TLS secures each user message separately, the overhead added for small messages is higher than for the other two security solutions where small messages are first bundled and then secured as a whole. This is the reason why the throughput of the TLS solution is significantly lower in this case compared to the other solutions allowing bundling. Considering the typically small size of signaling messages, this result is of particular interest if a security solution for signaling transport – the genuine SCTP application – has to be selected.

Figure 3 shows a scenario where the link is not the bottleneck of the transmission, so the throughput of the security solutions depends on the protocol, its implementation and the performance of the CPU. Such a scenario can be found in LANs, especially in todays 1 Gbps Ethernets. In this scenario where the host performance is the bottleneck, the cryptographic functions, in particular encryption, introduce a significant throughput penalty. This can be clearly seen by comparing the solutions to plain standard SCTP. Therefore, the ability to mix secured and unsecured data in one association is highly beneficial – favouring TLS over SCTP and S-SCTP. The measurement results show that the TLS solution achieves the highest throughput of all three security solutions for most packet sizes. The throughput of IPsec and S-SCTP is lower because the encryption of the data and

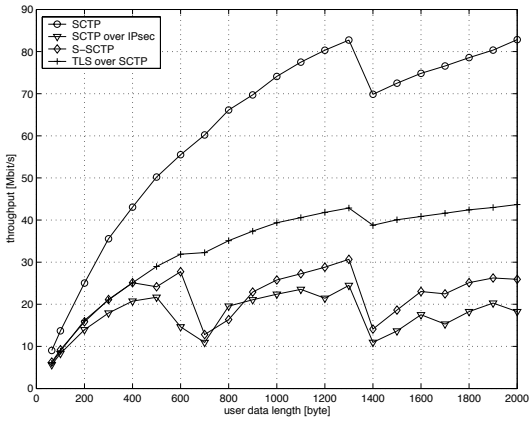

Figure 3. Throughput of security solutions in a 1Gbps Ethernet

the transmission occur in the same process. If the send queue of this process is full, the send call blocks and waits until new packets can be transmitted. During this time the process runs idle. In the case when bundling cannot be used any more (around 700 bytes of user data length) or when long messages have to be fragmented (1400 bytes of user data length) the throughput drops because there is more overhead contained in packets and the process cannot send more packets due to the blocking send call. When using TLS, encryption and transmission are handled by different processes. In this case, even if the send call blocks and waits until new packets can be transmitted, TLS can still encrypt data for future transmission.

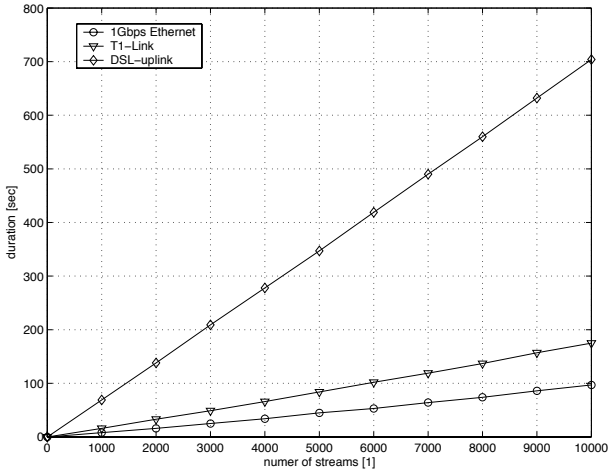

Figure 4. Duration to complete TLS handshakes

The last figure (Fig. 4) shows the time needed to establish TLS sessions over the given number of streams. As mentioned before, TLS has to establish a new secure session for each stream and as the measurements show, this can be very time consuming if many streams have to be secured or the link bandwidth is limited. Similar problems arise when TLS regularly performs a rekeying using an abbreviated handshake. This abbreviated handshake does not take as long as the inital handshake because there is no need to exchange certificates, but for a high number of streams this can also take several seconds to minutes, depending on the link.

From the measurements presented in this section and previous measurements [32] we can identify some limitations for the three security solutions. Over high bandwidth links, the throughput of IPsec is lower compared to the other solutions. If only a small portion of the transmitted data has to be secured, the disadvantage of IPsec is more severe because it can not differentiate between data that has to be secured and data that can be send unsecured. TLS over SCTP's major performance limitation is linked to the number of streams which have to be secured. With an increasing number of streams the memory usage and the time to establish the secure sessions for all streams also increase. A throughput degradation occurs when TLS has to secure small messages which can be bundled in the other solutions. S-SCTP was designed to overcome the performance limitations of SCTP over IPsec and TLS over IPsec, so we only identified some performance limitations using high bandwidth links. The main reason for this is the use of a prototype S-SCTP implementation developed to validate the design decisions which was not yet optimized for performance.

V. Two new security solutions: DTLS and SCTP-AUTH

A. DTLS

The new Datagram TLS (DTLS) is a modification of TLS allowing to support unreliable transport. It is defined in [18] and a good description is given in [37]. The main differences to TLS, which requires a reliable transport, are:

1) The DTLS messages not carrying user messages are retransmitted by the DTLS layer in case of packet loss.
2) The interdependence of successive TLS records is removed such that each received DTLS segment can be decrypted and handled independently.

These differences required changes in the packet format and the procedures of the protocol. DTLS is designed to secure UDP based communications but can also be used on top of the Datagram Congestion Control Protocol (DCCP) being defined in [17]. This usage is described in [22].

The simplest way of using DTLS in combination with SCTP is to just run plain DTLS over SCTP. This is currently done by IPFIX as described in [30]. It is clear that this solution can provide confidentiality and authentication of the user messages being transferred. However, the reliability of SCTP can not be provided to the user message transport. It is easy for an attacker to remove data chunks from the wire without interrupting the SCTP association. This stealing attack does not require that PR-SCTP is used. An attacker could just replace the user data of a data chunk with some older user data. The

receiving DTLS layer would just discard this old user message. It should be noted that this is not a problem in the IPFIX scenario: the IPFIX protocol is designed to run also on unreliable transport protocols like UDP.

Because of the strict separation of the layers all attacks against the transport layer to bring down an association like sending ABORT chunks, for example, are still possible. This is very similar to the TLS over TCP scenario.

B. SCTP-AUTH

The SCTP authentication mechanism described in [23] has been developed to overcome the security problems which were introduced to SCTP by the ADD-IP extension. These and other currently known attacks and their countermeasures are described in [25]. To solve the issue it was necessary for the receiver of ASCONF chunks used for reconfiguration to make sure that the chunks have not been modified and were sent by the SCTP endpoint which started the association.

During the establishment of the SCTP association the peers exchange 32-byte random numbers, a list of chunk types which have to be authenticated and the list of supported HMAC algorithms. It is possible that both sides have endpoint pair shared keys. It is out of scope of the SCTP-AUTH extension how these endpoint pair shared keys are exchanged. From the data being exchanged during the setup of the association and the endpoint pair shared keys the association shared keys are generated. These are then used to authenticate the chunks using the HMAC algorithm. SCTP-AUTH defines a new control chunk, called the AUTH chunk, which is used to transmit the HMAC of all chunks after the AUTH chunk contained in that packet.

If endpoint pair shared keys are used, an attacker can not insert chunks which are authenticated. If no endpoint pair shared keys are used, the attacker has to capture the association setup messages to be able to construct the association shared key.

Multiple endpoint based shared keys and therefore association shared keys are supported and identified by a key identifier. The endpoint based shared keys can be added, deleted and modified during the lifetime of an SCTP association by the application. In the socket API these modifications are done by socket options. See [28] for a detailed description.

The ADD-IP extension requires that the ADD-IP relevant chunk types ASCONF and ASCONF-ACK are always authenticated. But the SCTP-AUTH extension also allows a receiver to require that almost all chunk types have to be authenticated. This capability can be used to protect an SCTP association against an attacker, who wants to bring down an association. This usage is then similar to the TCP-MD5 extension being defined in [3].

The SCTP-AUTH extension is already implemented in the FreeBSD 7 source tree and there exist prototype implementations for the Linux SCTP kernel implementation.

VI. An alternative approach to SCTP end-to-end security

With S-SCTP, the major functional and performance issues associated with end-to-end security solutions for SCTP are – in principle – solved. However, broad acceptance and deployment would require full standardization in the IETF followed by providing S-SCTP kernel implementations for the major operating systems. One concern with respect to such kernel implementations is, that the kernel would have to perform some operations, like certificate verification and key establishment procedures, which could block the operating system for significant and unpredictable periods of time. Such a behaviour could compromise the responsiveness of the operating system and decrease its ability to handle real-time critical applications.

To strictly avoid this while still preserving the advantages of S-SCTP – in particular the capabilities to protect SCTP control traffic and to efficiently mix secured and unprotected traffic – we propose an alternative solution where the security functionality is split up. Encryption of data, data integrity and authentication are predictable and hence they can in principle be integrated into SCTP and implemented in the kernel. Session management, key management and user authentication using certificates on the other hand depend on factors that are not controllable by the operating system. For example user authentication depends on the user who has to present a valid certificate, additionally this certificate has to be checked. As a consequence, these functions have to be implemented in the user space and consequently above SCTP. Taking advantage of two IETF standardization efforts described above, namely DTLS and SCTP-AUTH, such a hybrid solution can be designed with minimal additional standardization impact.

DTLS can be used to support both the unordered delivery mode of SCTP as well as the SCTP extension for partial reliability. In addition, it also allows to use one common DTLS session for multiple SCTP streams avoiding the scalability problems with respect to the number of concurrent streams. The other weakness of TLS over SCTP, namely the inability to protect SCTP control traffic, has to be avoided by combining DTLS with the SCTP-AUTH extension.

There are two limitations of such a hybrid approach compared to S-SCTP:

1) It cannot provide confidentiality (encryption) for SCTP control chunks.
2) It does not allow a mixture of secure and insecure streams.

However, this is outweighed by the fact that neither additional changes to SCTP (which would be difficult to standardize) nor operating system kernel modifications are required.

With this combination of SCTP, DTLS and SCTP-AUTH, some modifications to DTLS are necessary. In addition, some functionality, e.g. replay protection and

reliable transport for secure session management information, can be provided at different levels. Therefore, the following section will describe the resulting solution called "SCTP aware DTLS" proposed by us in contrast to the simple SCTP unaware DTLS described in V-A in some more detail.

VII. Concept of SCTP aware DTLS

The functional block diagram of SCTP aware DTLS is shown in Fig. 5.

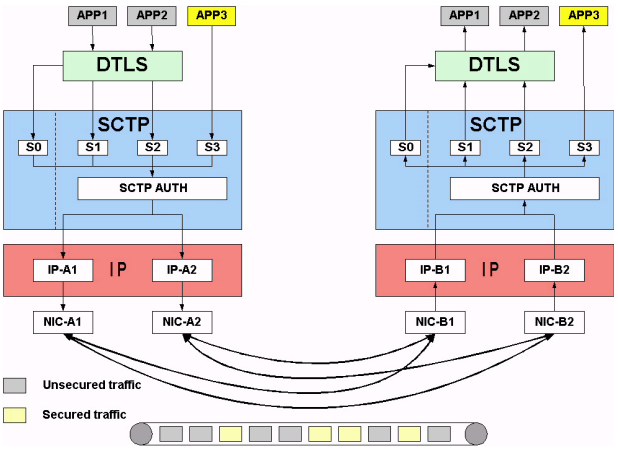

Figure 5. Concept of SCTP aware DTLS

If an application requires a secure end-to-end session, it first establishes an SCTP association. During the handshake both peers have to exchange some parameters regarding the SCTP-AUTH extension, including a list of chunks that are only accepted in an authenticated way. SCTP aware DTLS requires at least the authentication of all DATA, SACK (Selective Acknowledgement) and FORWARD-TSN chunks, the other chunk types (e.g. HEARTBEAT) can also be authenticated if required by the application using SCTP-AUTH. Initially an empty endpoint pair secret is used with the key identifier 0, later on the endpoint pair secret is derived from the master secret of the DTLS session by using the method described in [29]. Whenever DTLS changes the cipher spec a new endpoint pair secret is derived from the master secret using [29] and the key identifier is incremented.

When the application requests secure transport and triggers the handshake of the SCTP aware DTLS session it passes down the relevant details of the association which has to be protected. The handshake messages for session establishment and management are sent over stream 0, which is reserved for management traffic and provides reliable and ordered transport. Once a DTLS session is established, application data is protected by the DTLS security mechanisms and forwarded to the SCTP layer together with the SCTP specific control information (e.g. stream number).

If multiple applications (streams) require protection, they use the same DTLS session. Thus the scaling problem of TLS over SCTP is avoided. In the case an application does not need security, it can directly pass its data to SCTP. Even if no security is required, some chunks of this application will be authenticated by SCTP-AUTH, but this does not introduce any problems.

The usage of DTLS has one restriction regarding the message size an application can send over SCTP aware DTLS. A DTLS record only supports a maximum length of 2^{14} bytes of user data, all longer messages are rejected. At the moment there are no SCTP applications known which send longer messages, as long messages also introduce head-of-line blocking to SCTP.

A new issue is introduced in the case of a re-keying at the DTLS layer. The sender side DTLS has to buffer all new data that should be sent until it receives a notification, based on SCTP TSNs, that all data was received. Only then the sender can start the re-keying process and both sides can delete the old keying material. This method can cause a blocking effect among different streams since no new data is sent until the last message encrypted with the old keys is successfully transported and the re-keying was done. Since re-keying is not frequent, this is acceptable. The other method would be to keep old keying material in the case of a re-keying, but this would require a complex key management.

There are some optional features of DTLS which are unnecessary when DTLS is used over SCTP. First of all, the retransmission of DTLS control messages in the DTLS Handshake Layer is not necessary because they are transported in reliable mode by SCTP. The replay detection can be performed by SCTP in combination with SCTP-AUTH and is therefore not necessary at the DTLS layer. Since the replay detection at the DTLS layer might even result in dropping user messages it must not be used. The optional cookie exchange during DTLS session setup within the DTLS layer is not necessary because the SCTP association establishment procedure provides a similar service and is performed first.

A. Implementation considerations for SCTP aware DTLS

The concept of SCTP aware DTLS tries to keep the differences to standard DTLS as small as possible, in order to reuse the existing protocol infrastructure and implementation. This is beneficial when creating a secure and stable (prototype) implementation. Additionally, acceptance in the standardization process is easier if only small changes have to be made.

The current reference implementation of DTLS is based on the OpenSSL library. OpenSSL is an open source implementation of TLS which runs on all major operating systems. Since our previous TLS over SCTP and S-SCTP implementations are also based on OpenSSL, we can benefit from this experience when developing a prototype implementation. Figure 6 shows the structure of the modules in the DTLS implementation including the proposed modifications, the planned modifications are marked dark.

SCTP aware DTLS requires modifications to DTLS.

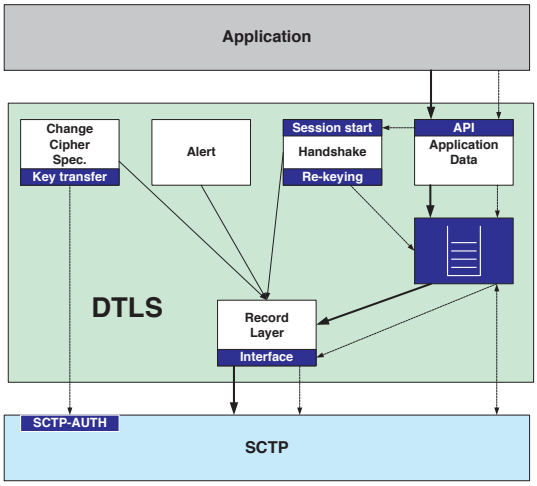

Figure 6. Structure of SCTP aware DTLS

If an application needs security, it starts a DTLS session and specifies the SCTP association which should be used to transport the data. When the first DTLS session is requested, the Handshake Layer starts the establishment of the secure session. During the handshake, the Record Layer binds the new session to the SCTP association specified by the application. For subsequent requests of other applications to open a secure session, DTLS can skip the handshake and signal the application that the session is ready for data transmission.

The Record Layer of DTLS is responsible for the transmission of Application Data, Alert, Handshake and Change Cipher Spec messages. If the Record Layer has to send application data it encrypts the data and uses the SCTP information provided by the application (e.g. stream number) to send it. All other messages are control messages and must be sent over stream 0 on the bound association.

The Handshake Layer is responsible for re-keying. However, before a re-keying can take place all outstanding data must be acknowledged. This can be checked by using a specific socket option of SCTP. Newly arriving messages from the application have to be buffered at the DTLS layer during this period. The additional buffering mechanism and its communication with both SCTP and the handshake module are some of the major adaptation efforts for the scheme.

Also all optional features of DTLS which are unnecessary when used over SCTP, are located in the Handshake Layer. Further analysis will show if it is beneficial to remove them or not.

In the Change Cipher Spec module only one modification must be made. When SCTP aware DTLS sends a Change Cipher Spec message, the new Master Secret must be passed down to the AUTH extension of SCTP and the key identifier must be incremented. There are no changes expected in the Alert protocol of DTLS.

B. API considerations for SCTP aware DTLS

There are two APIs involved in the implementation of DTLS over SCTP, on the one hand the API between the DTLS layer and the application layer, on the other hand the API between the DTLS layer and the SCTP layer.

Let us first consider the interface to the upper layer. One of the goals of our solution is that DTLS/SCTP provides the same services to the user as SCTP does. This means in particular that the user can specify the stream identifier, the payload protocol identifier, the time to live and possibly other parameters when sending messages. The existing interface between the user and the OpenSSL library consists of simple `SSL_write()` and `SSL_read()` calls as described in [38]. So for sending messages, one could still use the simple `SSL_write()` call and specify the above data by using `setsockopt()` of the standard SCTP socket API. However, when the user receives data with `SSL_write()` there is no way to transfer also the received payload protocol identifier, the stream identifier, etc. to the user. It is also not possible to provide any SCTP notifications to the user. This means that this API, at least for the receiving part, has to be extended. The simplest way would be to add a function called `SSL_sctp_write()` which has a similar signature as the `sctp_sendmsg()` function of the standard SCTP socket API.

Considering the interface between the DTLS library and the transport layer it is important to note that also only a very simple one is used which can just send or receive a message. SCTP, however, uses a far more complex interface. Not only the received user messages are delivered from SCTP to the application but also notifications. These are used, for example, to inform the user about state changes of the paths. However, these notifications have to be enabled explicitly by the user, but they are important for SCTP applications and therefore the applications running over DTLS/SCTP most likely want to enable them. Also additional information like the payload protocol identifier and the stream identifier are passed from the SCTP layer to the OpenSSL library. These are important for the application because the application might want to use the payload protocol identifier to demultiplex protocols as it is done in RSerPool. This means that the API for the receiving side has to be extended. For the sending side, either the OpenSSL library could be bypassed by using socket options directly or that interface has to be extended, too.

When using the socket API for SCTP, a decision has to be made whether to use the 1-to-1 style API or the 1-to-many style API. Since the programming model in OpenSSL is related to the one to one relation of BIO or SSL objects per TLS connection it seems more appropriate to use the 1-to-1 style API for SCTP.

VIII. CONCLUSION

Based on a comprehensive set of criteria we have evaluated the standard security solutions for SCTP and have identified their limitations. An optimized solution

solving these issues has been presented and lab tests based on a prototype implementation have confirmed the validity of the design choices. Acknowledging issues for the broad introduction of S-SCTP - which are mainly standardization related - we have proposed a new alternative. This "SCTP aware DTLS" solution preserves the advantages of S-SCTP while using emerging standard protocol components which hopefully increase the acceptance in the standardization groups. In addition to describing the overall concept of SCTP aware DTLS and its features in detail, we have also discussed the major aspects to be taken into account for the prototype implementation.

One standard component we use in our solution, DTLS [18], has already RFC status, the other, SCTP-AUTH [23], is expected to reach official RFC status soon. An Internet Draft [27], describing our proposed SCTP aware DTLS solution in detail is currently being discussed at the IETF. A prototype implementation using the OpenSSL library is under development.

REFERENCES

[1] J. Postel, *"Transmission Control Protocol"*, STD 7, RFC 793, September 1981.

[2] T. Dierks and C. Allen, *"The TLS Protocol"*, RFC 2246, January 1999.

[3] A. Heffernan, *Protection of BGP Sessions via the TCP MD5 Signature Option*, RFC 2385, August 1998.

[4] L. Ong, I. Rytina, M. Garcia, H. Schwarzbauer, L. Coene, H. Lin, I. Juhasz, M. Holdrege and C. Sharp, *"Framework Architecture for Signaling Transport"*, RFC 2719, October 1999.

[5] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, *"Stream Control Transmission Protocol"*, RFC 2960, October 2000.

[6] D. Mitton, M. St.Johns, S. Barkley, D. Nelson, B. Patil, M. Stevens and B. Wolff, *"Authentication, Authorization, and Accounting: Protocol Evaluation"*, RFC 3127, June 2001.

[7] M. Tuexen, Q. Xie, R. Stewart, M. Shore, L. Ong, J. Loughney and M. Stillman, *"Requirements for Reliable Server Pooling"*, RFC 3237, January 2002.

[8] L. Ong and J. Yoakum, *"An Introduction to the Stream Control Transmission Protocol (SCTP)"*, RFC 3286, May 2002.

[9] A. Jungmaier, E. Rescorla and M. Tuexen, *"Transport Layer Security over Stream Control Transmission Protocol"*, RFC 3436, December 2002.

[10] S. Bellovin, J. Ioannidis, A. Keromytis and R. Stewart, *"On the use of Stream Control Transmission Protocol (SCTP) with IPsec"*, RFC 3554, July 2003.

[11] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen and P. Conrad, *"Stream Control Transmission Protocol (SCTP) Partial Reliability Extension"*, RFC 3758, May 2004.

[12] J. Loughney, M. Tuexen and J. Pastor-Balbas, *"Security considerations for signaling Transport (SIGTRAN) Protocols"*, RFC 3788, June 2004.

[13] S. Kent and K. Seo, *"Security Architecture for the Internet Protocol"*, RFC 4301, December 2005.

[14] S. Kent, *"IP Authentication Header"*, RFC 4302, December 2005.

[15] S. Kent, *"IP Encapsulation Security Payload (ESP)"*, RFC 4303, December 2005.

[16] C. Kaufman, *"Internet Key Exchange (IKEv2) Protocol"*, RFC 4306, December 2005.

[17] E. Kohler, M. Handley, S. Floyd, *Datagram Congestion Control Protocol (DCCP)*, RFC 4340, March 2006.

[18] E. Resorla and N. Modadugu, *"Datagram Transport Layer Security"*, RFC 4347, April 2006.

[19] R. Stewart, I. Arias-Rodrigues, K. Poon, A. Caro and M.Tuexen, *Stream Control Transmission Protocol (SCTP) Specification Errata and Issues*, RFC 4460, April 2006.

[20] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka, *"Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration"*, draft-ietf-tsvwg-addip-sctp-20.txt (work in progress), April 2007.

[21] M. Riegel and M. Tuexen, *"Mobile SCTP"*, draft-riegel-tuexen-mobile-sctp-07.txt (work in progress), October 2006.

[22] T. Phelan, *Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)*, draft-ietf-dccp-dtls-00.txt (work in progress), May 2007.

[23] M. Tuexen, R. Stewart, P. Lei and E. Rescorla, *"Authenticated Chunks for Stream Control Transmission Protocol (SCTP)"*, draft-ietf-tsvwg-sctp-auth-08.txt (work in progress), February 2007.

[24] M. Stillman, R. Gopal, S. Sengodan E. Guttman and M. Holdrege, *"Threats Introduced by Rserpool and Requirements for Security in response to Threats"*, draft-ietf-rserpool-threats-06.txt (work in progress), November 2006.

[25] R. Stewart, M. Tuexen, G. Camarillo, *Security Attacks Found Against SCTP and Current Countermeasures*, draft-ietf-tsvwg-sctpthreat-03.txt (work in progress), April 2007.

[26] C. Hohendorf, E. Unurkhaan and T. Dreibholz, *"Secure SCTP"*, draft-hohendorf-secure-sctp-00.txt (work in progress), July 2005.

[27] M. Tuexen, C. Hohendorf, E. Rescorla, *Datagram Transport Layer Security for Stream Control Transmission Protocol*, draft-tuexen-dtls-for-sctp-01.txt (work in progress), October 2006.

[28] R. Stewart, Q. Xie, L. Yarroll, K. Poon and M. Tuexen, *Sockets API Extensions for Stream Control Transmission Protocol (SCTP)*, draft-ietf-tsvwg-sctpsocket-14.txt (work in progress), December 2006.

[29] E. Rescorla, *Keying Material Extractors for Transport Layer Security (TLS)*, draft-rescorla-tls-extractor-00.txt (work in progress), January 2007.

[30] B. Claise, *Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information*, draft-ietf-ipfix-protocol-24.txt (work in progress), November 2006.

[31] E. Unurkhaan, *"Secure End-to-End Transport — A new security extension for SCTP"*, Dissertation, University of Duisburg-Essen, June 2005.

[32] U. Esbold, E.P. Rathgeb and A. Jungmaier, *"Secure SCTP - A Versatile Secure Transport Protocol"*, Telecommunications 27:2-4, p.273ff, 2004.

[33] R. Stewart and Q. Xie, *"Stream Control Transmission Protocol - A Reference Guide"*, Addison-Wesley, 2002.

[34] W. R. Stevens, B. Fenner, A. Rudoff, *UNIX Network Programming*, Addison Wesley, 2004.

[35] A. Jungmaier, *"SCTP for beginners"*, http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb/, 2003.

[36] L. Rizzo, *"Dummynet"*, http://info.iet.unipi.it/~luigi/ip_dummynet/

[37] N. Modadugu and E. Resorla, *"The Design and Implementation of Datagram TLS"*, Network and Distributed System Security Symposium, February 2004.

[38] J. Viega, M. Messier and P. Chandra, *Network Security with OpenSSL*, O'Reilly, 2002.

[39] Carsten Hohendorf, Erwin P. Rathgeb, Esbold Unurkhaan, and Michael Tüxen, "*Secure End-to-End Transport over SCTP*", Proc. Emerging Trends in Information and Communication Security (ETRICS 2006), LNCS vol. 3995, 2006, 381-395.

Carsten Hohendorf was born in Oberhausen, Germany in 1976. He studied Computer Science and Communication Engineering at the University of DuisburgEssen and the University of Queensland in Brisbane, Australia. His main focus was mobile communication, especially multiantenna systems and computer networks. He received his Master degree in 2003 from the University of DuisburgEssen.

From 2004 to 2006, he was a member of the scientific staff at the Alfried Krupp von Bohlen und HalbachChair for "Computer Networking Technology" at the Institute for Experimental Mathematics, University of DuisburgEssen, Germany. During this time he worked on a project to secure the endtoend traffic of SCTP associations, which was founded by the German Research Foundation.

Since 2007 he is working at polypatent and started his studies to become a patent attorney.

Erwin P. Rathgeb was born in Ulm/Germany in 1958. He received his diploma and Ph.D. degrees in Electrical Engineering from the University of Stuttgart/Germany in 1985 and 1991, respectively. From 1985 to 1990, he was member of the scientific staff at the Institute of Communication Networks and Computer Engineering (Prof. Paul J. K¨uhn) at the University of Stuttgart where he was head of a research group on design and analysis of distributed systems.

From 1990 to 1991, he was a member of technical staff at Bellcore, Morristown, NJ/U.S.A., before joining Bosch Telekom in Backnang/Germany. In 1993, he joined Siemens in Munich/Germany. In various positions in systems engineering and product planning, he contributed to concepts for commercial ATM nodes and ATMbased multiservice networks. Since January 1999, he holds the Alfried Krupp von Bohlen und HalbachChair for "Computer Networking Technology" at the Institute for Experimental Mathematics, University of DuisburgEssen/Germany. He is the author of a book on ATM and has published more than 50 papers in journals and at international conferences. Professor Rathgeb is a senior member of IEEE and a member of GI, IFIP and ITG where he is chairman of the expert group on network security.

His current research interests include network security as well as concepts and protocols for next generation internets, in particular the Stream Control Transmission Protocol (SCTP) and Reliable Server Pooling.

Esbold Unurkhaan was born in Ulaanbaatar, Mongolia in 1975. He received his Bachelor degree and his Master degree in Computer Science from the Computer Science and Management School (CSMS) of the Mongolian University of Science and Technology (MUST) in 1997 and 1999, respectively. In 2005 he received his Ph.D. degree in Computer Science from the University of DuisburgEssen, Germany. From 1997 to 2001, he was a lecturer at the Computer Science and Management School of the Mongolian University of Science and Technology (MUST). From 2001 to 2004, he was member of the scientific staff at the Alfried Krupp von Bohlen und HalbachChair for "Computer Networking Technology" at the Institute for Experimental Mathematics, University of DuisburgEssen/Germany. After finishing his Ph.D., he returned to the CSMS. His current research interests focus on network security.

Michael Tüxen was born in 1966 in Oldenburg, Germany. He studied mathematics at the University of G ¨ottingen and received the Dipl. Math. degree in 1993. From 1993 to 1996 he was a member of the scientific staff at the Sonderforschungsbereich "Geometrie und Analysis" (SFB 170) in G¨ottingen. He received the Dr. rer. nat. degree in 1996. In 1997 he joined the Systems Engineering group of ICN WN CS of the Siemens AG in Munich. Since 2003 he is a professor at the Department for Electrical Engineering and Computer Science of the M¨unster University of Applied Sciences.

He was working in the Study Group 2 of the International Telecommunication Union (ITU) on performance analysis of signaling protocols.

At the Internet Engineering Task Force (IETF) he is active in the Working Groups Signaling Transport (SIGTRAN), Reliable Server Pooling (RSerPool), and Transport Area Working Group (TSVWG), where he is working on the design of protocols for signaling transport over IPnetworks and architectures for reliable distributed processing. He is an author of several Request for Comments (RFCs).

His current research interests include innovative transport protocol concepts, in particular the Stream Transmission Control Protocol (SCTP) and its implementation and applications.