

MPTCP  
Internet-Draft  
Intended status: Informational  
Expires: April 18, 2013

C. Paasch, Ed.  
O. Bonaventure  
UCLouvain  
October 15, 2012

MultiPath TCP Low Overhead  
draft-paasch-mptcp-lowoverhead-00

Abstract

This document describes a low overhead connection establishment mechanism for Multipath TCP. Its goal is to reduce the computational overhead of establishing an MPTCP connection and the associated TCP subflows in controlled environments where security attacks are not a concern.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Connection initiation . . . . .	3
3. Starting a new subflow . . . . .	6
4. Operation . . . . .	7
4.1. Generating the token . . . . .	7
4.2. Stateless Servers . . . . .	7
5. Security Considerations . . . . .	8
6. Informative References . . . . .	8
Authors' Addresses . . . . .	8

## 1. Introduction

This document introduces a variant of the MPTCP handshake that is suitable for an environment where security attacks are not an issue. The proposed handshake is a low overhead, low security version of the MPTCP handshake defined in [I-D.ietf-mptcp-multiaddressed].

Its goal is to provide an MPTCP handshake and authentication mechanism, reducing the computational overhead provided by MPTCP version 0.

## 2. Connection initiation

MultiPath TCP uses the MP\_CAPABLE option in the handshake for the initial subflow. This handshake was designed to meet several requirements. When designing another variant of the Multipath TCP handshake, it is important to have these requirements in mind. These requirements are :

1. Detect whether the peer supports MultiPath TCP.
2. Each host generates a locally unique token that unambiguously identifies the Multipath TCP connection
3. Agree on an Initial Data Sequence Number to initialize the MPTCP state on each direction of the Multipath TCP connection

Before discussing the proposed low overhead handshake, it is important to have in mind how [I-D.ietf-mptcp-multiaddressed] meets the three requirements above.

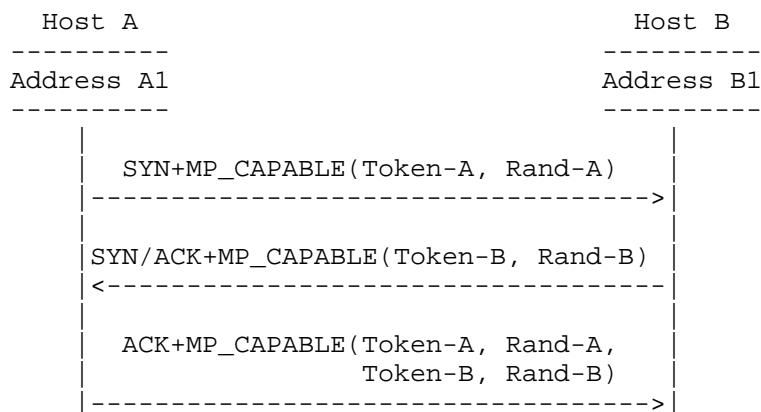
The first requirement is simply met by using a Multipath TCP specific option like all TCP extensions.

To meet the second requirement, a simple solution would have been to encode the token inside the MP\_CAPABLE option. However, this would have increased the size of the MP\_CAPABLE option. This would have limited the possibility of extending Multipath TCP later by adding new TCP options that require space inside the SYN segments. To minimize the number of option bytes consumed in the SYN segment, [I-D.ietf-mptcp-multiaddressed] uses a hash function to compute the token based on the keys exchanged in clear. However, using hash functions implies that implementations must handle the possible collisions which increases the complexity of the Multipath TCP handshake.

The third requirement is more subtle but is also important to ensure

the reliability of a Multipath TCP connection. Let us assume that Multipath TCP hosts do not agree on an Initial Data Sequence Number. Consider the following scenario. Host A opens the initial TCP subflow of the Multipath TCP connection. Host B opens a second subflow in this Multipath TCP connection. Host B sends one byte with DSN x over the initial subflow, but this data never reaches host A. Host B then sends one byte, starting at DSN x+1 over the second subflow. If host A does not know the Initial Data Sequence Number used by host B, it cannot determine whether the byte received over the second subflow can be acknowledged at the DSN level or not. [I-D.ietf-mptcp-multiaddressed] solves this problem by allowing the two hosts to derive the Initial Data Sequence Number from the keys exchanged in the MP\_CAPABLE option. However, this is achieved by computing a hash over the exchanged keys, which increases the computational overhead of generating/processing the MP\_CAPABLE option.

The figure below provides a simpler and low overhead handshake that meets the three requirements identified above.



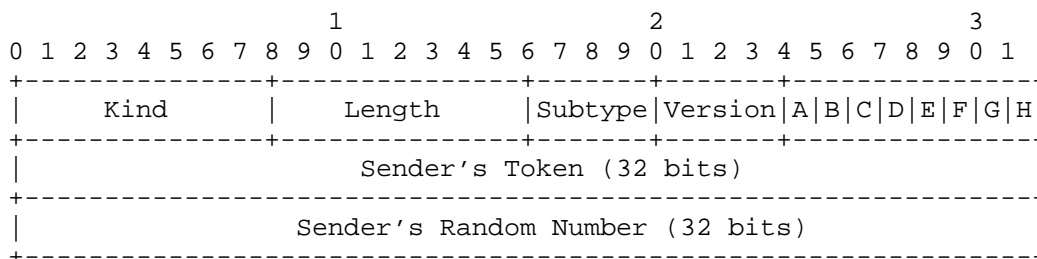
Handshake of the initial subflow.

Figure 1

MPTCP's establishment of the initial subflow follows TCP's regular 3-way handshake, but the SYN, SYN/ACK and ACK packets contain the MP\_CAPABLE-option. The proposed MP\_CAPABLE option contains one 32 bits token and one 32 bits random number in the SYN and SYN/ACK segments. The third ACK includes an MP\_CAPABLE option that contains the two tokens and random numbers. The tokens are used to explicitly exchange identifier of the Multipath TCP connection. The random numbers, combined with the tokens produce the Initial Data Sequence Numbers. Echoing all the information back in the third ACK allows

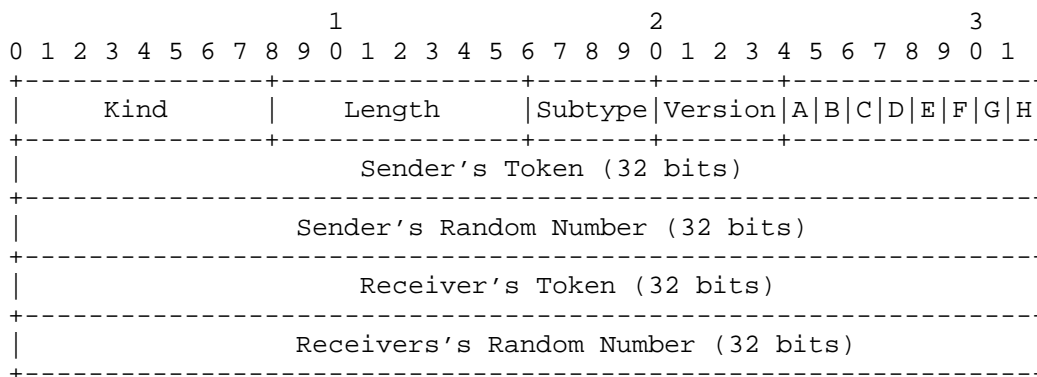
stateless operation of the server.

The format of the proposed MP\_CAPABLE option is proposed in the figures below.



Format of the MP\_CAPABLE-option in the SYN and SYN/ACK packets

Figure 2



Format of the MP\_CAPABLE-option in the third ACK of the handshake

Figure 3

The format of the MP\_CAPABLE option is shown in Figure 2. To indicate that this MP\_CAPABLE contains tokens/random numbers and not keys (as in [I-D.ietf-mptcp-multiaddressed], the Version-field is set to 1. The message format of the third ACK's MP\_CAPABLE option is show in Figure 3.

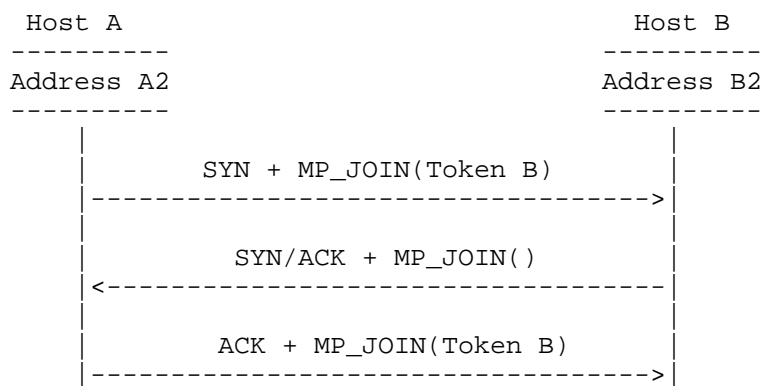
The Initial Data Sequence Number (IDSN) serves to initialize the MPTCP state on the end-hosts in the same way as TCP's sequence numbers do during the 3-way handshake. There is one IDSN for each direction of the data-stream. The IDSN for the data from the client

to the server is the concatenation of Rand-A and Token-A (Rand-A||Token-A). Rand-A is thus the high-order 32 bits of the IDSN, and Token-A the low-order 32 bits. For the data from server to client, the IDSN is the concatenation of Rand-B and Token-B (Rand-B||Token-B). Rand-A and Rand-B MUST be random numbers with sufficient randomness so that they are hard to guess. Recommendations for generating random numbers for use in keys are given in [RFC4086].

The meaning of the other fields and behavior of the end-hosts during the MP\_CAPABLE exchange is the same as specified in [I-D.ietf-mptcp-multiaddressed].

### 3. Starting a new subflow

Once an MPTCP connection has been established and the tokens exchanged, new subflows can be established. The establishment of the new subflows follows the handshake as show in Figure 4.

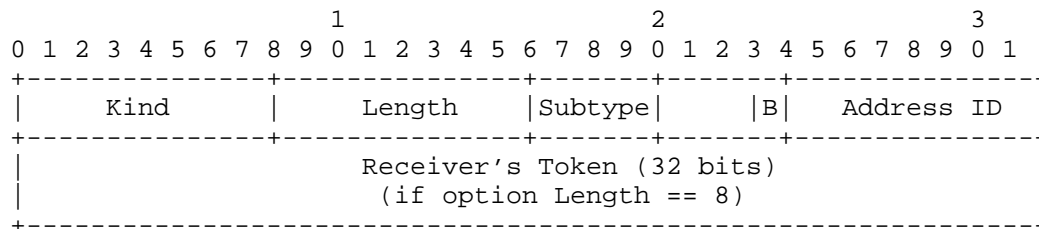


Handshake for a new subflow.

Figure 4

As the low-overhead version of MPTCP does not try to protect against hijacking attacks, the only goal of the MP\_JOIN inside the 3-way handshake is to identify the MPTCP connection this subflow is joining. The token inside the MP\_JOIN of the SYN-segment allows the server to identify the connection. The SYN/ACK also contains an MP\_JOIN option because the server needs to signal to the client that it indeed received the SYN together with the MP\_JOIN and that there is no middlebox that removes MPTCP options on this path. Finally, the client replies with the third ack. This third ack contains again token B. This allows the server to handle MP\_JOIN's in a stateless manner, as described below. The third ack is sent in a reliable

manner as explained in [I-D.ietf-mptcp-multiaddressed].



Format of the MP\_JOIN-option

Figure 5

The semantics of the backup-bit "B" and the Address ID are the same as in [I-D.ietf-mptcp-multiaddressed].

## 4. Operation

### 4.1. Generating the token

The token must only be locally unique. The method used to generate the token is implementation specific. One possible way to generate the token is by applying a block-cipher on a counter together with a local secret. This approach has the benefit of a higher probability of uniqueness of the token. We will only have a token collision after the counter has wrapped around. This means, that a connection must have survived  $2^{32}$  other connections to cause a collision. Thus, a token collision is less likely to occur than with [I-D.ietf-mptcp-multiaddressed].

### 4.2. Stateless Servers

To allow stateless SYN+Join handling, the server has to perform the following upon reception of a SYN:

- o Check whether there exists an MPTCP-connection corresponding to the token inside the MP\_JOIN option.
- o Send a SYN/ACK as it is done on today's stateless servers.

When receiving the third ACK (sent reliably as it is done in today's MPTCP), the server verifies that indeed it has generated a SYN/ACK (like regular TCP's SYN-cookie mechanism) and thanks to the token echoed back in the third ACK, the server can find the MPTCP-session this subflow is joining.

Handling the SYN+Join in a stateless manner allows the server to protect itself against attackers that are flooding the server with SYN+Join messages. As the server does not need to create state when sending the SYN/ACK, flooding performed by the attacker will not prevent real clients from establishing new subflows.

## 5. Security Considerations

The proposed solution removes the HMAC authentication mechanism described in [I-D.ietf-mptcp-multiaddressed]. It is assumed that end-hosts will only use this low-overhead version of MPTCP for non-security critical traffic or in controlled environments like isolated data-centers.

Security-critical traffic is nowadays typically sent over SSL/TLS or similar secure application level protocols. This is done because the transport protocols like TCP do not provide a sufficient security. An application using SSL over MPTCP benefits from the same security provided by SSL. There is one downside of using SSL over MPTCP. If an attacker manages to join an existing connection thanks to a JOIN-exchange, he can inject data into the SSL-session. However, thanks to the MAC-authentication of the SSL messages, the end-hosts will tear down the SSL session.

## 6. Informative References

- [I-D.ietf-mptcp-multiaddressed]  
Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,  
"TCP Extensions for Multipath Operation with Multiple  
Addresses", draft-ietf-mptcp-multiaddressed-10 (work in  
progress), October 2012.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness  
Requirements for Security", BCP 106, RFC 4086, June 2005.

## Authors' Addresses

Christoph Paasch (editor)  
UCLouvain  
Place Sainte Barbe, 2  
Louvain-la-Neuve, 1348  
BE

Email: christoph.paasch@uclouvain.be

Olivier Bonaventure  
UCLouvain  
Place Sainte Barbe, 2  
Louvain-la-Neuve, 1348  
BE

Email: [olivier.bonaventure@uclouvain.be](mailto:olivier.bonaventure@uclouvain.be)



MPTCP  
Internet-Draft  
Intended status: Informational  
Expires: April 18, 2013

C. Paasch, Ed.  
O. Bonaventure  
UCLouvain  
October 15, 2012

Securing the MultiPath TCP handshake with external keys  
draft-paasch-mptcp-ssl-00

Abstract

Multipath TCP currently relies on the exchange of keys in clear during the initial handshake to authenticate the establishment of additional subflows. This document proposes a variant of the Multipath TCP handshake that allows Multipath TCP to reuse keys negotiated by the Application layer protocol above it such as SSL/TLS to authenticate the establishment of additional subflows.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

## Table of Contents

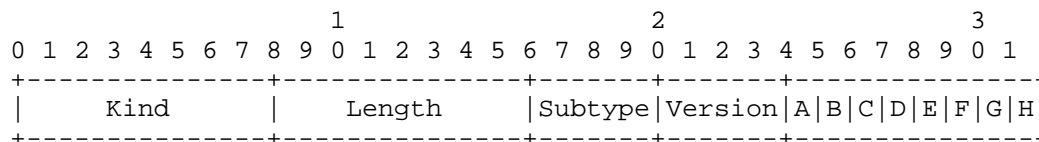
1. Introduction . . . . .	3
2. Connection initiation . . . . .	3
3. Multipath TCP API . . . . .	4
4. Starting a new subflow . . . . .	4
5. Deployment . . . . .	6
6. Security Considerations . . . . .	7
7. Informative References . . . . .	7
Authors' Addresses . . . . .	7

## 1. Introduction

Multipath TCP is an extension to TCP that enables hosts to use multiple paths to exchange data for a single connection. [I-D.ietf-mptcp-multiaddressed] describes the current design of the Multipath TCP protocol. The design of Multipath TCP has been influenced by various factors including the backward compatibility with regular TCP, the fallback to TCP when middleboxes interfere with the Multipath TCP options, ... The design of Multipath TCP has also been affected by security requirements. The security threats against Multipath TCP are documented in [RFC6181]. Multipath TCP aims at being no worse than TCP from a security viewpoint. Other approaches such as [I-D.bittau-tcp-crypt] or [RFC5925] have been proposed to reduce the vulnerability of TCP to attacks. Multipath TCP currently addresses the security threats identified in [RFC6181] by exchanging keys during the handshake for the initial subflow. These keys are then used to generate HMACs to authenticate the establishment of subsequent TCP subflows. Exchanging keys in clear during the initial handshake has obvious shortcomings from a security viewpoint. However, some application-layer protocols like SSL/TLS or ssh already negotiate a shared key between the end-points. In this document we propose a modification to the handshake used by Multipath TCP for the initial and subsequent subflows that enables Multipath TCP to rely on an application-supplied key to authenticate the establishment of the subflows.

## 2. Connection initiation

The handshake of the initial subflow is a small variation to the handshake of [I-D.ietf-mptcp-multiaddressed] or draft-paasch-mptcp-lowoverhead-00. The header of the MP\_CAPABLE option of these two MPTCP-versions has the format as shown in the below figure.



Header of the MP\_CAPABLE option

Figure 1

We propose to use the B bit in this option to indicate whether the host that sent the MP\_CAPABLE option will use an application supplied key to authenticate the additional subflows or not. When the B bit

is set, it indicates that the authentication key is supplied by the application. If the B bit has not been set in both directions, the authentication mechanism is used as defined by the MPTCP version ([I-D.ietf-mptcp-multiaddressed] or draft-paasch-mptcp-lowoverhead-00).

In MPTCP version 0, even if the B bit is set the end-hosts still have to generate a key that fulfills the requirements as defined in MPTCP version 0. This is necessary to handle the case where the client supports the B bit, but the server not yet. For a more in-depth analysis of this kind of deployment scenario, have a look at Section 5.

By using the same handshake as draft-paasch-mptcp-lowoverhead-00, the proposed handshake can also benefit from the lower overhead for generating the token and thus the faster establishment of the initial subflow.

### 3. Multipath TCP API

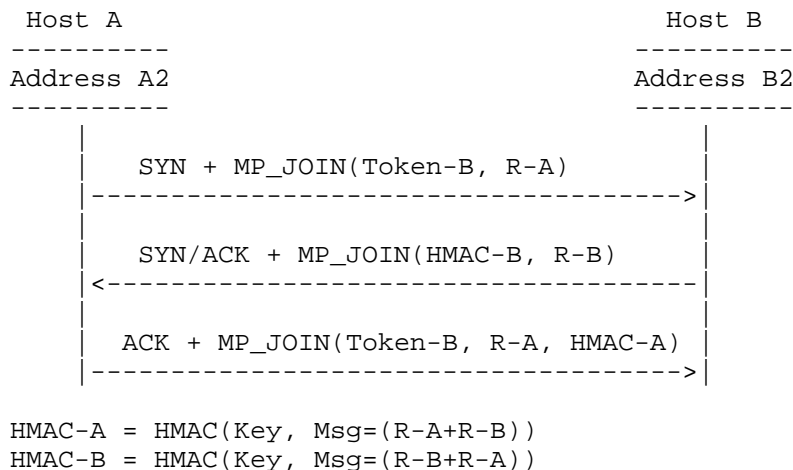
The proposed mechanism requires an interaction between the application and the MPTCP layer. This can be achieved by the means of socket options. Two socket options are necessary:

- o MPTCP\_ENABLE\_APP\_KEY : This socket option tells the socket layer that an application supplied key will be used to secure the establishment of additional subflows. This socket option MUST be used before establishing the initial subflow, or before starting to listen on a socket to accept new connections. When this socket option is used, the MP\_CAPABLE option is sent with the "B"-bit set to 1.
- o MPTCP\_KEY : This socket option allows the application to provide a key to the MPTCP layer. Both end-points MUST use this socket option in order to allow the MPTCP-layer to create new subflows. It is up to the application to negotiate the key between the end-points. E.g., in the case of SSL/TLS, the key can be a hash of the shared secret that has been negotiated with the SSL exchange. Separate documents will describe in details how applications such as TLS or SSH can pass a shared secret to Multipath TCP by using this option.

### 4. Starting a new subflow

The handshake for the establishment of a new subflow is similar to the one specified in [I-D.ietf-mptcp-multiaddressed]. There are two

important differences. First, the HMAC is computed by using the keys provided by the application. Second, the token and the client's random number are included inside the third ack to allow stateless operation of the passive opener of an additional subflow.

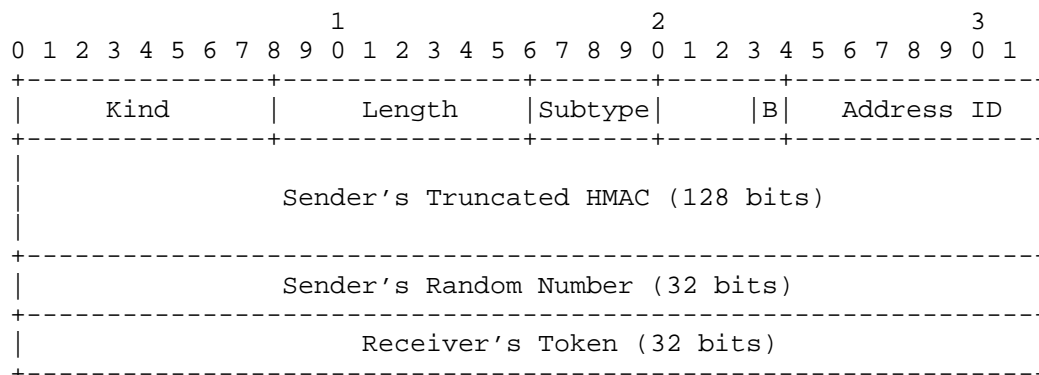


Handshake of a new subflow.

Figure 2

In order to allow the Token-B and R-A inside the third ack, the HMAC-A must also be a truncated version of the 160-bit HMAC-SHA1. Thus, HMAC-A is the truncated (leftmost 128 bits) of the HMAC as shown in Figure 2.

The message-format of the MP\_JOIN-option in the SYN and the SYN/ACK is the same as in [I-D.ietf-mptcp-multiaddressed]. As the third ACK includes the Token and the random nonce, the MP\_JOIN message format of the third ack is as show in Figure 3. The length of the MP\_JOIN-option in the third ACK is 28 bytes. There remains thus enough space to insert the timestamp option in the third ACK.



Format of the MP\_JOIN-option

Figure 3

The semantics of the backup-bit "B" and the Address ID are the same as in [I-D.ietf-mptcp-multiaddressed].

## 5. Deployment

This proposed mechanism assumes that the application uses new socket-options to provide the key to the MPTCP-layer. Thus, the first requirement for deploying this MPTCP handshake is that the TLS/SSL-layer has been modified. There may of course be scenarios, where the client is supporting the proposed solution, but the server not. Thus, the client sends out the MP\_CAPABLE with the B bit set, but the server replies without enabling the B bit. Upon reception of the SYN/ACK, it is up to the client's policy how to react. It can either continue with the negotiated version of MPTCP but without using the key from the application or fallback to regular TCP.

The applications will have to pass the shared key to the MPTCP-layer by the means of a socket-option. It may be that the client's application has already done the call to the socket-option but the server's application not yet. The server will receive a SYN with the MP\_JOIN-option, without knowing the key. In that case the server should silently drop the SYN. The TCP retransmission mechanism on the client-side will retransmit the SYN after the initial RTO expired (after 1 second). And the server's application potentially will have finally set the key via the socket-option.

## 6. Security Considerations

It is recommended that the applications do not pass the plain shared key to the MPTCP layer. They should rather pass a hash of their shared secret to the MPTCP layer. These security considerations will be discussed in documents that describe how applications such as TLS/SSL or SSH can interact efficiently with Multipath TCP.

## 7. Informative References

[I-D.bittau-tcp-crypt]

Bittau, A., Boneh, D., Hamburg, M., Handley, M., Mazieres, D., and Q. Slack, "Cryptographic protection of TCP Streams (tcpcrypt)", draft-bittau-tcp-crypt-03 (work in progress), September 2012.

[I-D.ietf-mptcp-api]

Scharf, M. and A. Ford, "MPTCP Application Interface Considerations", draft-ietf-mptcp-api-05 (work in progress), April 2012.

[I-D.ietf-mptcp-multiaddressed]

Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", draft-ietf-mptcp-multiaddressed-10 (work in progress), October 2012.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, March 2011.

## Authors' Addresses

Christoph Paasch (editor)  
UCLouvain  
Place Sainte Barbe, 2  
Louvain-la-Neuve, 1348  
BE

Email: christoph.paasch@uclouvain.be

Olivier Bonaventure  
UCLouvain  
Place Sainte Barbe, 2  
Louvain-la-Neuve, 1348  
BE

Email: [olivier.bonaventure@uclouvain.be](mailto:olivier.bonaventure@uclouvain.be)

