Overview

- Issues with TLS
- Proposal
- Performance comparison
- Open-questions in the proposal
Issues with TLS

- If the CBC-ciphersuites are implemented by the book/RFC are vulnerable to attacks [0,1]
Issues with TLS

- If the CBC-ciphersuites are implemented by the book/RFC are vulnerable to attacks [0,1]
- There are known attacks in RC4 that cannot be mitigated [2]
Issues with TLS

- If the CBC-ciphersuites are implemented by the book/RFC are vulnerable to attacks [0,1]
- There are known attacks in RC4 that cannot be mitigated [2]

Issues with TLS

- RC4 cannot be used in Datagram TLS (DTLS)
Issues with TLS

- RC4 cannot be used in Datagram TLS (DTLS)
  - No stream ciphers in DTLS
Issues with TLS: result

- That leaves us with few options
  - AES-GCM
    - Very fast on certain CPUs
    - Decent performance otherwise
That leaves us with few options

- **AES-GCM**
  - Very fast on certain CPUs
  - Decent performance otherwise

- **AES-CCM**
  - Decent performance
Issues with TLS: result

- That leaves us with few options
  - AES-GCM
    - Very fast on certain CPUs
    - Decent performance otherwise
  - AES-CCM
    - Decent performance
  - Both are only applicable to TLS 1.2+ or DTLS 1.2+
Issues with TLS: result

- When decent performance isn’t enough, a **fast** and **secure** stream cipher is needed
Proposal

- We propose to use the eStream [0] results to define a fast stream cipher for TLS/DTLS.

[0]. The eSTREAM project was a multi-year effort, running from 2004 to 2008, to promote the design of efficient and compact stream ciphers suitable for widespread adoption. As a result of the project, a portfolio of stream ciphers was announced in April 2008 and revised in 2012.
We propose to use the eStream [0] results to define a fast stream cipher for TLS/DTLS

- ESTREAM-SALSA20-HMAC-SHA1
- SALSA20-HMAC-SHA1
Proposal

- We propose to use the eStream [0] results to define a fast stream cipher for TLS/DTLS
  - ESTREAM-SALSA20-HMAC-SHA1
  - SALSA20-HMAC-SHA1
- and also utilize a fast MAC algorithm
  - ESTREAM-SALSA20-UMAC
  - SALSA20-UMAC
We propose to use the eStream [0] results to define a fast stream cipher for TLS/DTLS
- ESTREAM-SALSA20-HMAC-SHA1
- SALSA20-HMAC-SHA1

and also utilize a fast MAC algorithm
- ESTREAM-SALSA20-UMAC
- SALSA20-UMAC

UMAC as in RFC4418 (UMAC-AES)
Performance comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mbyte/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTREAM-SALSA20-SHA1</td>
<td>77</td>
</tr>
<tr>
<td>ESTREAM-SALSA20-UMAC</td>
<td>130</td>
</tr>
<tr>
<td>RC4-SHA1</td>
<td>61</td>
</tr>
<tr>
<td>AES-128-CBC-SHA1</td>
<td>45</td>
</tr>
<tr>
<td>AES-128-GCM</td>
<td>44</td>
</tr>
</tbody>
</table>
Packet Overhead

- Packet overhead per ciphersuite (in DTLS):

<table>
<thead>
<tr>
<th>Ciphersuite</th>
<th>Overhead</th>
<th>% of 1500</th>
<th>Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128-CBC-HMAC-SHA1</td>
<td>50-65</td>
<td>3.3-4.3</td>
<td>13 + 20 (MAC) + 16 (IV) + 16 (PAD)</td>
</tr>
<tr>
<td>AES-128-GCM</td>
<td>37</td>
<td>2.4</td>
<td>13 + 16 (MAC) + 8 (IV)</td>
</tr>
<tr>
<td>SALSA20-256-HMAC-SHA1</td>
<td>33</td>
<td>2.2</td>
<td>13 + 20 (MAC)</td>
</tr>
<tr>
<td>SALSA20-256-UMAC96</td>
<td>25</td>
<td>1.6</td>
<td>13 + 12 (MAC)</td>
</tr>
</tbody>
</table>
Packet Overhead

- Packet overhead per ciphersuite (in DTLS):

<table>
<thead>
<tr>
<th>Ciphersuite</th>
<th>Overhead</th>
<th>% of 1500</th>
<th>Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128-CBC-HMAC-SHA1</td>
<td>50-65</td>
<td>3.3-4.3</td>
<td>13 + 20 (MAC) + 16 (IV) + 16 (PAD)</td>
</tr>
<tr>
<td>AES-128-GCM</td>
<td>37</td>
<td>2.4</td>
<td>13 + 16 (MAC) + 8 (IV)</td>
</tr>
<tr>
<td>SALSA20-256-HMAC-SHA1</td>
<td>33</td>
<td>2.2</td>
<td>13 + 20 (MAC)</td>
</tr>
<tr>
<td>SALSA20-256-UMAC96</td>
<td>25</td>
<td>1.6</td>
<td>13 + 12 (MAC)</td>
</tr>
</tbody>
</table>

the packet counter is the nonce
Open-questions in proposal

- UMAC can be used
  - As in RFC4418 (UMAC-AES)
  - Or with the combined cipher (i.e., Salsa20)
Open-questions in proposal

- UMAC can be used:
  - As in RFC4418 (UMAC-AES)
  - Or with the combined cipher (i.e., Salsa20)

- Poly1305 is another option for a MAC
  - With comparable speed
  - Proposed in 2005 (UMAC in 1999)
  - No RFC
Conclusion

- We can have a replacement of RC4 that is:
  - More secure (one of the winners in eStream competition)
  - Faster
    - 2x-3x the speed of AES ciphersuites
    - 2x the speed of RC4 when combined with UMAC
  - Can be used efficiently with DTLS
Questions and Discussion
Salsa20 cryptanalysis

- Mouha, Nicky, and Bart Preneel. "A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis."
- Pelissier, Sylvain. "Cryptanalysis of Reduced Word Variants of Salsa." Western European Workshop on Research in Cryptology, WEWoRC. Vol. 44. 2009.
- Estream portfolio page for Salsa20: http://www.ecrypt.eu.org/stream/e2-salsa20.html
Performance comparison (full)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Throughput (Mbyte/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTREAM-Salsa20-SHA1</td>
<td>77</td>
</tr>
<tr>
<td>ESTREAM-Salsa20-UMAC</td>
<td>130</td>
</tr>
<tr>
<td>Salsa20-SHA1</td>
<td>69</td>
</tr>
<tr>
<td>Salsa20-UMAC</td>
<td>108</td>
</tr>
<tr>
<td>RCA-SHA1</td>
<td>61</td>
</tr>
<tr>
<td>AES-128-CBC-SHA1</td>
<td>45</td>
</tr>
<tr>
<td>AES-128-GCM</td>
<td>44</td>
</tr>
</tbody>
</table>

*Note: Throughput values are approximate and may vary.*