Zero-Knowledge Proofs meet TLS

Paul Grubbs
based on joint work with:
Arasu Arun, Joseph Bonneau,
Zachary DeStefano, Michael Walfish, Ye Zhang

paulgrubbs12@gmail.com
Twitter: @pag_crypto
TLS is awesome

TLS 1.3 is great, and it’s being deployed all over the place. Congrats!
"Visibility" Challenges

Network operators enforce security policies by scanning traffic...

DNS filtering
Data loss prevention
Intrusion detection
Malware scanning...

By design, TLS prevents scanning!
[TLS] Industry Concerns about TLS 1.3

BITS Security <BITSSecurity@fsroundtable.org> | Thu, 22 September 2016 17:24 UTC | Show header

To: IETF TLS 1.3 Working Group Members

My name is Andrew Kennedy and I work at BITS, the technology policy division of the Financial Services Roundtable (http://www.fsroundtable.org/bits). My organization represents approximately 100 of the top 150 US-based financial services companies including banks, insurance, consumer finance, and asset management firms.

The Network Is Going Dark: TLS 1.3 and Security Operations Visibility

Addressing Visibility Challenges with TLS 1.3
Proposed Solutions

Prior work:
- multi-context TLS (mcTLS)
- ETS/"enterprise TLS"
- trusted hardware

Keep scanning, other functionality in network...

...using weakened, modified TLS.
Zero-Knowledge Proofs meet TLS

• Talk about recent work: “Zero-Knowledge Middleboxes”
  • Key contribution: zero-knowledge decryption of TLS records
• ZKP+TLS in general. Future directions
• Solicit feedback, criticism, suggestions for use cases
Zero-Knowledge Proofs

Prover wants to convince verifier the statement is true, without revealing why.

Prover generates zero-knowledge proof, sends to verifier.

Verifier checks , learns the statement is true but not prover’s "reasoning"
Zero-Knowledge Middleboxes (ZKMBs)

1. Clients get policy on network join

2. Client+server do regular handshake to derive key

3. Client enforces policies locally, issues *zero-knowledge proof* that plaintexts are compliant

4. Middlebox acts as verifier, blocks traffic if verify fails

ZKMBs enforce policies on traffic they cannot see.
ZKMB example:
DNS-over-TLS/HTTPS (DoT/DoH) filtering

(1) Clients get domain blocklist on network join

(2) Client + DoT server do TLS 1.3 handshake to derive key

(3) Client issues zero-knowledge proof that domain name in DNS query is not on blocklist. Reveals nothing about its DNS query!

(4) Middlebox acts as verifier, blocks traffic if verify fails
Key challenge

(1) Clients get domain blocklist on network join

(2) Client + DoT server do TLS 1.3 handshake to derive key

(3) Client issues zero-knowledge proof that domain name in DNS query is not on blocklist. Reveals nothing about its DNS query!

(4) Middlebox acts as verifier, blocks traffic if verify fails

Need efficient ZK circuit for decrypting TLS 1.3 records. ZKMB will compose it with other ZK circuits for policy checking (see paper)
Decryptiong TLS 1.3 Records in ZK

How to build ZK circuit that decrypts a TLS 1.3 record?

Problem: TLS 1.3 records are **not binding**: can have multiple correct decryptions. Attacker can “lie in the proof” about its message.
Decrypting TLS 1.3 Records in ZK

How to build ZK circuit that decrypts a TLS 1.3 record?

Idea to fix: prove key was output from handshake

How do we build this key consistency check for TLS 1.3?
Key Consistency Check for TLS 1.3
Key Consistency Check for TLS 1.3
(the short version)

• Simple, inefficient: re-run most of client’s key derivation in circuit.
  • Diffie-Hellman values are binding to shared secret
• Observation: handshake ”commits to” intermediate steps of key derivation. Check these to shortcut key derivation
Experimental Results

- Key consistency check (one-time cost per TLS session): \(~15\text{ seconds}\)
- Cost for DoT filtering use case: \(~3\text{ seconds per DNS query}\)
- Proof verification cost: \(5\text{ milliseconds}\)
- tl;dr: not practical yet, but close.
  - Many possible optimizations – ongoing work reduces client costs by 70x, with caveats.
  - See paper online for more numbers
Zero-Knowledge Proofs meet TLS

• Talk about recent work: “Zero-Knowledge Middleboxes”
  • Key contribution: zero-knowledge decryption of TLS record
• ZKP+TLS in general. Future directions
• Solicit feedback, criticism, suggestions for use cases
ZKP+TLS future directions

- ZKPs are an intriguing new tool. Many interesting applications to TLS.
  - ZKPs of Encrypted Client Hello (ECH) plaintexts
- Open problems, future directions as well.
  - Key updates, client authentication, HRR, PHA...
  - Explicit support for ZKPs in TLS via extensions?
- Outlandish suggestion: could design future versions of TLS to be "ZKP-friendly"
Conclusion

• TLS deployments cause “visibility” challenges in networks. ZKPs can help.
• Described new research on *zero-knowledge middleboxes*, which use ZKPs to enable privacy-preserving enforcement of network policies like DNS filtering
  • https://eprint.iacr.org/2021/1022
• Many remaining research opportunities in ZKP+TLS. Interesting design consideration for future TLS versions.

Thanks for listening! Any questions?

paulgrubbs12@gmail.com
Twitter: @pag_crypt0
Experimental Results

Results: (* = ChaCha20 instead of AES)

### Channel Opening Microbenchmark

<table>
<thead>
<tr>
<th>CO Method</th>
<th>#Gates x 10^4</th>
<th>Time (s)</th>
<th>SRS (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>748</td>
<td>94.0</td>
<td>1200</td>
</tr>
<tr>
<td>Shortcut</td>
<td>111</td>
<td>16.5</td>
<td>149</td>
</tr>
<tr>
<td>Early data</td>
<td>61</td>
<td>8.6</td>
<td>81</td>
</tr>
</tbody>
</table>

### DNS Case Studies (after setup)

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Ctxt Size</th>
<th>#Gates x 10^4</th>
<th>Time (s)</th>
<th>SRS (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoT*</td>
<td>255</td>
<td>19.5</td>
<td>3.1</td>
<td>32</td>
</tr>
<tr>
<td>DoH</td>
<td>500</td>
<td>49.5</td>
<td>6.8</td>
<td>75</td>
</tr>
<tr>
<td>ODoH</td>
<td>500</td>
<td>48.0</td>
<td>6.4</td>
<td>61</td>
</tr>
</tbody>
</table>

Prototype can generate proof for nontrivial ZKMB in 3 seconds.