Enhancing Security and Privacy of Tor’s Ecosystem by using Trusted Execution Environments

Seongmin Kim, Juhyeng Han, Jaehyeong Ha,
Taesoo Kim *, Dongsu Han
Tor anonymity network

- Tor: the most popular anonymity network for Internet users
  - Helps users to defend against traffic analysis and keep user’s privacy (e.g., what sites you visit, IP address) [from Tor project, www.torproject.org]
  - Freely available as an open source
  - 1.8 million users on a daily basis

The geographic location of Tor relays *

* from Onionview, https://onionview.codeplex.com/
Tor anonymity network

- Tor: the most popular anonymity network for Internet users
  - Helps users to defend against traffic analysis and keep user’s privacy (e.g., what sites you visit, IP address) [from Tor project, www.torproject.org]
  - Freely available as an open source
  - 1.8 million users on a daily basis

The geographic location of Tor relays *

* from Onionview, https://onionview.codeplex.com/
Tor anonymity network

- Tor: the most popular anonymity network for Internet users
  - Helps users to defend against traffic analysis and keep user’s privacy (e.g., what sites you visit, IP address)  [from Tor project, www.torproject.org]
  - Freely available as an open source
  - 1.8 million users on a daily basis

The geographic location of Tor relays *

Not anonymous: attack reveals BitTorrent users on Tor network

An ingenious attack by French researchers has found a way to identify ...
Tor anonymity network

- Tor: the most popular anonymity network for Internet users
  - Helps users to defend against traffic analysis and keep user’s privacy (e.g., what sites you visit, IP address) [from Tor project, www.torproject.org]
  - Freely available as an open source
  - 1.8 million users on a daily basis

The geographic location of Tor relays *

Tor Project Offers a Secure, Anonymous Journalism Toolkit

"One cell is enough to break Tor's anonymity"

Not anonymous: attack reveals BitTorrent users on Tor network

https://onionview.codeplex.com/
Tor network: Threat model

- 3-hop onion routing: a single Tor entity cannot know both client and server
Tor network: Threat model

• 3-hop onion routing: a single Tor entity cannot know both client and server

• Tor’s Threat model
  – Tor is a volunteer-based network: Tor relays are not trusted

  Can run a Tor relays of his own
  Can compromise some fraction of Tor relays
Tor network: Threat model

- 3-hop onion routing: a single Tor entity cannot know both client and server

- Tor’s Threat model
  - Tor is a volunteer-based network: Tor relays are not trusted

- Can run a Tor relay of his own
- Can compromise some fraction of Tor relays
- Can observe some fraction of network traffic
Tor network: Threat model (Cont.)

- Careful admission
- Behavior monitoring
Tor network: Threat model (Cont.)

- Careful admission
- Behavior monitoring

Anonymity Broken!
Out-of-scope: network-level adversary (controls a large fraction of network)

1. Currently runs ~10,000 relays
2. Large-scale traffic correlation is believed to be verify difficult in practice
Tor network: Threat model (Cont.)

- Careful admission
- Behavior monitoring

Directory authorities

Tor client

Destination

Anonymity Broken!

• Having a large number of relays

However, Tor is still vulnerable to many types of attacks under its traditional threat model
Limitations of Tor

Problem 1. Tor relays are semi-trusted
   – Authorities cannot fully verify the behaviors of them

Problem 2. Even attackers control a few Tor relays, they can
   – Access internal information (circuit identifier, cell header, ...)
   – Modify the behavior of relays (DDoS, packet tampering, ...)

<Low-resource attacks>

- Malicious circuit creation [Security09, CCS11]
- Sniper attack [NDSS15]
- Bad apple attack [LEET11]
- Tagging attack [ICC08, TON12, CCS12, S&P13]
- Bandwidth inflation [PETS07, S&P13]
- Controlling HSDir [S&P13]
- Harvesting hidden service descriptors [S&P13]
- Circuit demultiplexing [S&P06]
- Website fingerprinting [Security15]
Limitations of Tor (Cont.)

Tor clients

Entry

Middle

Exit

Destination

Processing Unit: Cell (512 Bytes)

Information visible to attackers

Cell: CircID | CMD | DATA

header

Demultiplex and identify a circuit

Attacker can modify the behavior

Modify or inject the cell

Give false information to others

Bandwidth

20MB/s

150MB/s

Inflated!
To address the problems on Tor,

1) Fundamental trust bootstrapping mechanism
2) Advanced trust model to verify untrusted remote parties are required
Trend: Commoditization of TEE

- Trusted Execution Environment (TEE): Hardware technology for trusted computing

  - Intel SGX: a promising TEE technology for generic applications
    - Native performance in the secure mode
    - Available on Intel Skylake and Kaby lake CPU

  - Integrity checking → Prevents behavior modification
  - Cannot access data, flow control X → Protects the secrecy of the program
SGX-Tor: Leveraging Intel SGX on Tor

**Improved trust model**
- Spells out what users trust in practice
- Provides ultimate privacy

**Operational privacy**
- Protects sensitive data and Tor operations
- Prevents modifications on Tor relays

**Practicality**
- The chance of having more hardware resources donated
- Incrementally deployable
- Compatibility
SGX-Tor: Leveraging Intel SGX on Tor

- Reduces the power of an attacker who currently gets the sensitive information by running Tor relays

- Raises the bar for Tor adversary to a traditional network-level adversary (only passively see the TLS bytestream)
Intel SGX 101: Isolated Execution

- Protects app’s secret from untrusted privilege software
- Application keeps its data/code inside the “Enclave”
- Trusted Computing Base (TCB) = Enclave + CPU package
Intel SGX 101: Remote attestation

• Attest an application on remote platform
  – Checks the **integrity of enclave** (hash of code/data pages)
  – Verifies whether **enclave is running on real SGX CPU**
  – Can establish a “**secure channel**” between enclaves

![Diagram of remote attestation process]
SGX-Tor: Threat Model

Only trusts the underlying SGX hardware & Tor code itself

Do not address network-level adversaries: who can perform large-scale traffic analysis

Out of scope: Vulnerabilities in Tor codes, SGX side channel attacks

Mitigated by recent SGX research: Moat [CCS16], SGX-Shield [NDSS17], T-SGX [NDSS17]
SGX-Tor: Design and Implementation

User process (Tor application)

Enclave memory

Tor code/data (Core)
- Circuit Establishment
- Hidden service
- Encryption/Decryption
- Cell/Consensus creation

SSL Library
Integrity check with remote host

Attestation Module
Seals/unseals private data

Sealing Module

Securely obtains the entropy and time value
Validates the enclave hash of the Tor program
Encrypts and stores the sensitive data outside the enclave
SGX-Tor: Design and Implementation

User process (Tor application)

Enclave memory

- Tor code/data (Core)
  - Circuit Establishment
  - Hidden service
  - Voting
  - Encryption/Decryption
  - Cell/Consensus creation

- SSL Library
  - Integrity check with remote host

- Attestation Module
  - Seals/unseals private data

- Sealing Module

Application memory

- SGX Runtime Library
- OCALL/ECALL Wrapper
- Tor code/data (Untrusted)
- Standard Library (glibc)

Enclave initialization

OCALL

ECALL

Request system services

Trusted

Untrusted

Crypto/TLS operations

System Call

Enclave Creation

Enclave initialization
SGX-Tor: Design and Implementation

User process (Tor application)

Enclave memory

- Tor code/data (Core)
  - Circuit Establishment
  - Hidden service
  - Voting
  - Encryption/Decryption
  - Cell/Consensus creation

- SSL Library
  - Integrity check with remote host

- Attestation Module
  - Seals/unseals private data

- Sealing Module

Application memory

- SGX Runtime Library
- OCALL/ECALL Wrapper

- Tor code/data (Untrusted)
  - Standard Library (glibc)

OCALL

Request system services

ECALL

Enclave initialization

Narrow interface

Sanity checking
1. Argument length
2. Address range

Crypto/TLS operations

System Call

Trusted

Untrusted
Attacks defeated by using SGX-Tor

1. Tagging attack

2. Bandwidth inflation

2.1. BW scanning
2.2. Detect scanning
2.3. Report fake BW

Directory authorities
4. Create consensus document

Malicious relay (modified Tor)

160
120
80
40
0
Inflated!
Advertised BW
Attacks defeated by using SGX-Tor

1. Tagging attack
2. Bandwidth inflation

1. BW scanning
2. Detect scanning
3. Report fake BW
4. Create consensus document

Advertised BW
Inflated!

Directory authorities

Attract more clients!
Attacks defeated by using SGX-Tor (Cont.)

Attacks defeated/mitigated by SGX-Tor
- Circuit demultiplexing [S&P06]
- Bandwidth inflation [PETS07, S&P13]
- Harvesting/Controlling HSDir [S&P13]
- Tagging attack [ICC08, TON12, CCS12, S&P13]

Attacker cannot access:
1. Circuit identifier
2. Cell header
3. Private keys

modify the code:
1. To modify/inject cells
2. To inflate bandwidth

Enclave
- Onion/SSL key creation
- Circuit establishment
- Cell creation
- Encryption/Decryption

Private keys
- Circuit descriptor
- Cell

Attackers cannot access:
1. Circuit identifier
2. Cell header
3. Private keys

modify the code:
1. To modify/inject cells
2. To inflate bandwidth
New functionality: Automatic admission

- Integrity verification of relays (Directory authority → Onion Router)
  - Automatically admits "unmodified" and "SGX-enabled" relays
  - **Improved trust model**: current implicit trust model turns into the explicit trust model

**NOTE**: Tor uses the same binary for directory authorities, Tor relays, and client proxies
Incremental deployability

• **SGX-Tor’s basic assumption:** “All relays and authorities are SGX-enabled”

• **SGX-Tor supports interoperability**
  – Allows admission of non-SGX relays without remote attestation
  – SGX-enabled clients can get the list of SGX-Tor relays from SGX-enabled authorities
Implementation detail

• Engineering efforts
  – Support for Windows/Linux (based on Intel SGX SDK)
  – SGX-ported libraries: OpenSSL, libevent, zlibc
  – SGX-Tor is an open source: Available at https://github.com/KAIST-INA/SGX-Tor

• Trusted Computing Base (TCB) size
  – 3.8x smaller (320K LoC vs 1,228K LoC) than Graphene (open source library OS for SGX)
Evaluation

1) What kind of sensitive data of Tor is protected by SGX-Tor?
2) What is the performance overhead of running SGX-Tor?
3) How compatible and incrementally deployable is SGX-Tor with the current Tor network?

• Environmental setup
  – SGX CPUs: Intel Core i7-6700 (3.4GHz) and Intel Xeon CPU E3-1240 (3.5GHz)
  – Configuration: 128MB Enclave Page Cache (EPC)
  – Running Tor in Windows, Firefox as a Tor browser (in the client proxy)
  – Establish a private Tor network using chutney
### What is protected by SGX-Tor?

<table>
<thead>
<tr>
<th></th>
<th>Current Tor</th>
<th>Network-level adversary</th>
<th>SGX-Tor</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP header</td>
<td>Visible</td>
<td>Visible</td>
<td>Visible</td>
</tr>
<tr>
<td>TLS-encrypted bytestream</td>
<td>Visible</td>
<td>Visible</td>
<td>Visible</td>
</tr>
<tr>
<td>Cell</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
<tr>
<td>Circuit ID</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
<tr>
<td>Voting result</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
<tr>
<td>Consensus document</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
<tr>
<td>Hidden service descriptor</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
<tr>
<td>List of relays</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
<tr>
<td>Private keys</td>
<td>Visible</td>
<td>Not visible</td>
<td>Not visible</td>
</tr>
</tbody>
</table>
Performance evaluation

• SGX-Tor performance: WAN setting
  – Establish a private Tor network
  – For the realistic scenario, we consider the “locality of relays” (Asia, EU, U.S. West, U.S. East)

<Evaluation environment>

<Graphs>
- Throughput (Mbps)
  - File Size (MB): 10MB, 100MB
  - Protocols: HTTP, HTTPS
  - SGX-Tor vs Original Tor
  - 11.9% degradation (on average)

- Time-to-first-byte (ms)
  - Cumulative Prob.
  - File Size (MB): 10MB, 100MB
  - Protocols: HTTP, HTTPS
  - Enclave
  - 3.9% additional latency
Performance evaluation (Cont.)

- End-to-end client performance of SGX-Tor (using Tor browser)
  - Web latency: Visiting Alexa Top 50 websites
  - Hidden service: HTTP file server (downloading 10MB)

![Graph showing webpage loading time and hidden service throughput for SGX-Tor and Original Tor.]

**Webpage Loading Time (s)**
- SGX-Tor: 13.2s
- Original: 12.2s
- 7.4% additional latency

**Hidden Service Throughput (Mbps)**
- SGX-Tor: 1.30Mbps
- Original: 1.35Mbps
- 3.3% degradation
Compatibility with vanilla Tor

• Long-running: Admit SGX-Tor relays in the vanilla Tor
  – Collected results for two weeks

  Network I/O bandwidth per second
  Middle selection Probability
  Advertised bandwidth *

  Serves Tor traffic well
  Actually selected by multiple Tor users
  Listed in the consensus document

* From https://collector.torproject.org/
Conclusion

• We design and implement SGX-Tor by leveraging commodity TEE and demonstrate its viability
  – Gives moderate performance overhead
  – Shows its compatibility and possibility of incremental deployment

• SGX-Tor enhances the security and privacy of Tor by
  – Defending against existing attacks on Tor
  – Bringing changes to the trust model of Tor
  – Providing new properties: automatic admission

• Available at github! (https://github.com/KAIST-INA/SGX-Tor)