Practical and Provably Secure Onion Routing
[IEEE CSF '12]

Michael Backes
Saarland University
MPI-SWS

Ian Goldberg
University of Waterloo

Aniket Kate
MPI-SWS

Esfandiar Mohammadi
Saarland University
Anonymous Web Browsing

Alice → AIDS → Google
Anonymous Web Browsing
Anonymous Web Browsing

Alice

AIDS

We should talk.

Google

AIDS

Insurance
Anonymous Web Browsing

What happened now?

Alice

Google

We should talk.

Insurance
Anonymous Web Browsing

What happened now?

Next time I anonymize my search via Tor.

We should talk.

Practical and Provably Secure OR - Esfandiar Mohammadi
Anonymous Web Browsing

Tor

anonymizing
Anonymous Web Browsing: Tor

- established system for anonymous web browsing
- But: analyzing Tor is challenging
Tor: An Onion Routing Network
Anonymity against a malicious server
Anonymity against a malicious server

Phase 1: Establish Circuit
Anonymity against a malicious server

Phase 1: Establish Circuit

$k_1$ $cid_1$
**Tor: An Onion Routing Network**

Phase 1: Establish Circuit

- $k_1 \ cid_1$
- $k_2 \ cid_2$

Anonymity against a malicious server
**Tor: An Onion Routing Network**

**Phase 1: Establish Circuit**

- $k_1 \ cid_1$
- $k_2 \ cid_2$
- $k_3 \ cid_3$

Anonymity against a malicious server
Phase 1: Establish Circuit

Anonymity against a malicious server
Practical and Provably Secure OR - Esfandiar Mohammadi

Tor: An Onion Routing Network

Phase 2: Send message

\[ cid_1 || E(k_1, E(k_2, E(k_3, E(k_4, m)))) \]

Anonymity against a malicious server
Tor: An Onion Routing Network

$cid_1 || E(k_1, E(k_2, E(k_3, E(k_4, m))))$

Anonymity against a malicious server
Anonymity against a malicious server

Practical and Provably Secure OR - Esfandiar Mohammadi

Tor: An Onion Routing Network

\[ cid_2 \parallel E(k_2, E(k_3, E(k_4, m))) \]
Anonymity against a malicious server

Practical and Provably Secure OR - Esfandiar Mohammadi

Tor: An Onion Routing Network
Anonymity against a malicious server

\[ E(k_4, m) \]

Practical and Provably Secure OR - Esfandiar Mohammadi
Tor: An Onion Routing Network

Anonymity against a malicious server
Tor: An Onion Routing Network

Anonymity against a malicious server
Tor: An Onion Routing Network

• analyzing tor is difficult
• typical approach
  – abstract OR as a black-box
Tor: An Onion Routing Network

• analyzing tor is difficult
• typical approach

But does such a black-box abstraction capture all attacks?
Our Contribution

- We introduce a comprehensive black-box for onion routing
  - We bridge the gap between a known black-box abstraction and the onion routing (OR) protocol
- Our result even holds in the presence of universal composability (UC)
- We apply our result by introducing a definition for forward secrecy
  - We make a first step towards proving forward secrecy
Our Contribution

• We introduce a comprehensive black-box for onion routing
  – We bridge the gap between a known black-box abstraction and the onion routing (OR) protocol
• Our result even holds in the presence of universal composability (UC)
• We apply our result by introducing a definition for forward secrecy
  – We make a first step towards proving forward secrecy
Our Contribution

- We introduce a comprehensive black-box for onion routing
  - We bridge the gap between a known black-box abstraction and the onion routing (OR) protocol
- Our result even holds in the presence of universal composability (UC)
- We apply our result for forward secrecy
  - We make a first step towards proving forward secrecy

UC ➔ Timing attacks are not covered
Outline

- Recall UC
- Discussing our Black-Box
- Challenges with the current Tor protocol
- Main Result & Applications
Universal Composability

Attacker models:
- Compromised protocol parties
- Compromised network links
Universal Composability

Trusted Party (black-box) computing the result of P modelling honest parties

calling protocol ↔
calling protocol

calling protocol
Universal Composability

Trusted Party
(black-box)
computing the result of P
modelling honest parties
Universal Composability

Comparing two worlds
Universal Composability

∀ \exists \approx

trusted party (black-box) computing the result of P modelling honest parties
Universal Composability

∀

∃

Trusted Party
(black-box)
computing the result of P
modelling honest parties

calling protocol

protocol P

trusted party

protocol P

trusted party

calling protocol
Universal Composability

protocol $P$

trusted party (black-box) computing the result of $P$
modelling honest parties

A ⊗ E ⊗ A
Universal Composability

protocol \( P \)

trusted party (black-box) computing the result of \( P \) modelling honest parties

\( \exists \forall \)

A

E

A
An elegant black-box for OR

\[ b : \text{fraction of compromised nodes} \]

\[
\begin{align*}
S & \\
\text{user} & \rightarrow
\begin{cases}
  b^2 & x := (U, S) \\
  (1-b)^2 \cdot b & x := (\neg, S) \\
  b \cdot (1-b) & x := (U, \neg) \\
  (1-b)^2 \cdot (1-b) & x := (\neg, \neg)
\end{cases}
\text{with probability}
\end{align*}
\]

\[ x \rightarrow \text{attacker} \]

by Feigenbaum, Johnson, and Syverson (to appear in TISSec)
An elegant black-box for OR

Why is this not sufficient?

With probability

\[ b \cdot b \quad x := (U,S) \]
\[ (1-b) \cdot b \quad x := (-,S) \]
\[ b \cdot (1-b) \quad x := (U,-) \]
\[ (1-b) \cdot (1-b) \quad x := (-,-) \]

by Feigenbaum, Johnson, and Syverson (to appear in TISSec)
An elegant black-box for OR

Why is this not sufficient?

- abstraction is sound if no circuit is reused

\[
\begin{align*}
\text{user} \rightarrow S \\
\text{with probability} \\
\begin{align*}
(b \times b) & \quad x := (U,S) \\
((1-b) \times b) & \quad x := (-,S) \\
(b \times (1-b)) & \quad x := (U,-) \\
((1-b) \times (1-b)) & \quad x := (-,-)
\end{align*}
\]

\[b : \text{fraction of compromised nodes}\]

by Feigenbaum, Johnson, and Syverson (to appear in TISSec)
How does the simulation work?
The simulator

$b$ : fraction of compromised nodes

with probability

\[
\begin{align*}
  b * b & : x := (U,S) \\
  (1-b) * b & : x := (-,S) \\
  b * (1-b) & : x := (U,-) \\
  (1-b) * (1-b) & : x := (-,-)
\end{align*}
\]
Reusing Circuits
Reusing Circuits
Reusing Circuits

This circuit has been reused.
Reusing Circuits

This circuit has been reused.
This circuit has been reused.
Reusing Circuits
Reusing Circuits (Overapproximation)

If $h = -$ draw a fresh handle $h$
else check whether $h$ is valid

with probability

\[
\begin{align*}
    b \cdot b & \quad x := (h, U, S, m) \\
    (1-b) \cdot b & \quad x := (h, -, S, m) \\
    b \cdot (1-b) & \quad x := (h, U, -) \\
    (1-b) \cdot (1-b) & \quad x := (h, -, -)
\end{align*}
\]

$b$ : fraction of compromised nodes

$(S, m, h)$

user

$x$ ::= $(h, U, S, m)$

$x$ ::= $(h, -S, m)$

$x$ ::= $(h, U, -)$

$x$ ::= $(h, -, -)$

attacker
The simulator

$b$ : fraction of compromised nodes

If \( h = - \) draw a fresh handle \( h \)
else check whether \( h \) is valid

with probability

\[
\begin{align*}
    &b \cdot b & x &= (h, U, S, m) \\
    & (1-b) \cdot b & x &= (h, -, S, m) \\
    &b \cdot (1-b) & x &= (h, U, -) \\
    &(1-b) \cdot (1-b) & x &= (h, -, -)
\end{align*}
\]

If \( h \) is not known
establish a circuit \( C \)
else look up \( C \)
Send \( m \) to \( S \) over \( C \)
The simulator

\( b \) : fraction of compromised nodes

If \( h = - \) draw a fresh handle \( h \)
else check whether \( h \) is valid

with probability:

\[
\begin{align*}
    b \cdot b \\
    (1 - b) \cdot b \\
    b \cdot (1 - b) \\
    (1 - b) \cdot (1 - b)
\end{align*}
\]

\( x := (h, -, S, m) \)

\( x := (h, U, -) \)

\( x := (h, -, -) \)

\( (S, m, h) \)

Drawback: we leak the reusage of a circuit via \( h \)

If \( h \) is not known
establish a circuit \( C \)
else look up \( C \)

Send \( m \) to \( S \) over \( C \)
Reusing Circuits (tighter)

$$S$$

user

draw a random circuit
draw a handle \((h,P,Q)\) for every ciphertext between \(P, Q\)
leak these handles

attacker
A different Corruption Scenario

This circuit has been reused
The simulator

\[(m, S)\]
The simulator

Black-box

$(h, P_1, P_2, P_3)$

$(m, S)$
The simulator

Black-box

\( (h, P_1, P_2, P_3) \)

\( E(k_2, E(k_3, 0^l)) \)

\( (m, S) \)
The simulator

Black-box

\[ E(k_2, E(k_3, 0^l)) \]

\[ (h, P_1, P_2, P_3) \]

\[ (m, S) \]
The simulator

Black-box

\( (m, S) \)

\( E(k_3, 0^l) \)

\( (h, P_1, P_2, P_3) \)
The simulator

Black-box

\( (m, S) \)

\( (h, P_1, P_2, P_3) \)

\( h \)

\( E(k_3, 0^l) \)
Practical and Provably Secure OR - Esfandiar Mohammadi

The simulator

Black-box

(h, P1, P2, P3)

(m, S)

h
The simulator

Black-box

\[(m, S)\]

\[(h, P_1, P_2, P_3)\]

\[(m, P_3, P_4, S)\]

\[h\]
The simulator

Black-box

\[ E(k_4, m) \]

\[ (h, P_1, P_2, P_3) \]

\[ (m, P_3, P_4, S) \]
The simulator

\[ E(k_4, m) \]

\[ (h, P_1, P_2, P_3) \]

\[ (m, P_3, P_4, S) \]

\[ (m, S) \]
The simulator

Black-box

\[ (m, S) \]

\[ (m, P_3, P_4, S) \]

\[ (h, P_1, P_2, P_3) \]

\[ h \]

\[ m \]
Our Black-Box

• Allows reusing circuits
  – Performs circuit construction
  – Maintains circuits
  – Draws a fresh handle for every ciphertext
Are we done?
Are we done?

• How to prove circuit creation secure?
  – Goldberg, Stebila, and Ustaoglu introduced an efficient & secure one-way AKE [DCC].
  – Perfectly suited for our proof.
Are we done?

• How to prove circuit creation secure?
  – Goldberg, Stebila, and Ustaoglu introduced an efficient & secure one-way AKE [DCC].
  – Perfectly suited for our proof.

• For non-malleable encryption schemes that's it.
Are we done?

• How to prove circuit creation secure?
  – Goldberg, Stebila, and Ustaoglu introduced an efficient & secure one-way AKE [DCC].
  – Perfectly suited for our proof.

• For non-malleable encryption schemes that's it.

• But Tor uses the malleable detCTR scheme:
Are we done?

• How to prove circuit creation secure?
  – Goldberg, Stebila, and Ustaoglu introduced an efficient & secure one-way AKE [DCC].
  – Perfectly suited for our proof.
• For non-malleable encryption schemes that's it.
• But Tor uses the malleable detCTR scheme:
  \[ E(k, m) \oplus c = E(k, m \oplus c) \]
A Problem with Malleability

\[ E(k_1, E(k_2, E(k_3, E(k_4, m)))) \]
A Problem with Malleability

\[ E(k_1, E(k_2, E(k_3, E(k_4, m)))) \]
A Problem with Malleability

\[ E(k_2, E(k_3, E(k_4, m))) \]
A Problem with Malleability

\[ E(k_2, E(k_3, E(k_4, m))) \oplus c \]
A Problem with Malleability

$$E(k_2, E(k_3, E(k_4, m \oplus c)))$$
A Problem with Malleability

\[ E(k_3, E(k_4, m \oplus c)) \]
A Problem with Malleability

$E(k_4, m \oplus c)$
A Problem with Malleability

$m \oplus c$

$m \oplus c$

$m \oplus c$
Recall what we have to prove

protocol P

protocol P

Trusted Party
(black-box)
computing the result of P
modelling honest parties

∀ A ∈ E A
A Problem with Malleability

$(m, S)$
A Problem with Malleability

Black-box

\[(h, P_1, P_2, P_3)\]

\[(m, S)\]
A Problem with Malleability

\[ E(k_2, E(k_3, 0^l)) \]

\( (m, S) \)

Black-box
A Problem with Malleability

Black-box

\( (m, S) \)

\( E(k_2, E(k_3, 0^l)) \)

\( (h, P_1, P_2, P_3) \)
A Problem with Malleability

Black-box

\[(h, P_1, P_2, P_3)\]

\[E(k_2, E(k_3, 0^l)) \oplus c\]

\[(m, S)\]
A Problem with Malleability

$\Pr(k_2, E(k_3, 0^l \oplus c))$

$(h, P_1, P_2, P_3)$

$(m, S)$

Black-box
A Problem with Malleability

Black-box

\[ E(k_3, 0^l \oplus c) \]

\[(h, P_1, P_2, P_3)\]

\[(m, S)\]
A Problem with Malleability

Black-box

$E(k_3, 0^l \oplus c)$

$(h, P_1, P_2, P_3)$
A Problem with Malleability

Black-box

(h, P_1, P_2, P_3)

h

(m, P_3, P_4, S)

↑

(m, S)
A Problem with Malleability

Black-box

\[ (m, S) \]

\[ (h, P_1, P_2, P_3) \]

\[ h \]

\[ (m, P_3, P_4, S) \]

\[ E(k_4, m) \]
A Problem with Malleability

Black-box

\[ E(k_4, m) \]

\( (h, P_1, P_2, P_3) \)

\( h \)

\( (m, P_3, P_4, S) \)

\( (m, S) \)
A Problem with Malleability

Black-box

\[(m, P_3, P_4, S)\]

\[h\]

\[(h, P_1, P_2, P_3)\]

\[m \neq m \oplus c\]
Predictable Malleability

• It turns out:
  We can predict the changes in the plaintext.

• There is a poly-time $S$ and $T$ s.t.
  $$S(w, w') = T \quad \text{and} \quad T(m) = D(k, w')$$

• Generalized: Predictable Malleability
  for stateful encryption schemes (details in the paper)

⇒ Remedy: black-box additionally allows the simulator to send a transformation $T$
A Problem with Malleability

(\(m, S\))
A Problem with Malleability

Black-box

$(h, P_1, P_2, P_3)$

$(m, S)$
A Problem with Malleability

\[ E(k_2, E(k_3, 0^l)) \]

\[ (h, P_1, P_2, P_3) \]
A Problem with Malleability

Black-box

\[ E(k_2, E(k_3, 0^l)) \]

\[ (h, P_1, P_2, P_3) \]

\[ (m, S) \]
A Problem with Malleability

Black-box

\[(h, P_1, P_2, P_3)\]

\[E(k_2, E(k_3, 0^l)) \oplus c\]
A Problem with Malleability

\[ E(k_2, E(k_3, 0^l \oplus c)) \]

Black-box

\[ (m, S) \]
A Problem with Malleability

Black-box

\[ E(k_3, 0^l \oplus c) \]

\[ (h, P_1, P_2, P_3) \]
A Problem with Malleability

Black-box

\[ E(k_3, 0^l \oplus c) \]

\[ (h, P_1, P_2, P_3) \]

\[ T(m) := m \oplus c \]

\[ (m, S) \]
A Problem with Malleability

\( T(m) := m \oplus c \)

\((h, P_1, P_2, P_3)\)

\((h, T)\)

\((T(m), P_3, P_4, S)\)

\((m, S)\)
A Problem with Malleability

\[ T(m) := m \oplus c \]

Black-box

\[ (h, P_1, P_2, P_3) \]

\[ (h, T) \]

\[ (T(m), P_3, P_4, S) \]
A Problem with Malleability

Black-box

\[ T(m) := m \oplus c \]

\[ E(k_4, m \oplus c) \]

\[ (h, P_1, P_2, P_3) \]

\[ (h, T) \]

\[ (T(m), P_3, P_4, S) \]
A Problem with Malleability

Black-box

\[ T(m) := m \oplus c \]

\[ (h, P_1, P_2, P_3) \]

\[ (h, T) \]

\[ (T(m), P_3, P_4, S) \]
Our main result

• The Tor protocol
  – allowing to reuse circuits
  – with a strengthened integrity check
  – with secure one-way AKE
  – against (partially) global active attackers
    (details in the paper)

• realizes our black-box
Applications of this result

• For the OR anonymity analysis of Feigenbaum, Johnson, and Syverson
  – we show the exact conditions under which their result applies.

• We did a first step towards proving forward secrecy
Future Work

• Incorporate **timing attacks** into the analysis
• Is a black-box leaking the **reusage** of a circuit useful?
• Implications of **removing the TLS link** between routers
  – the **circuit ids** are leaked to a network attacker
• Predictable malleability might be a useful notion for simulation-based proofs
  – e.g., for protocols that use efficient but malleable stream ciphers
Recall the elegant Overapproximation

\[ b : \text{fraction of compromised nodes} \]

\[(S, m, h)\]

If \( h = - \) then

- draw a fresh handle \( h \)

with probability

\[ b * b \]

\[ (1-b) * b \]

\[ b * (1-b) \]

\[ (1-b) * (1-b) \]

user \[\rightarrow\]

attacker
Future Work

• Incorporate timing attacks into the analysis
• Is a black-box leaking the reusage of a circuit useful?
• Implications of removing the TLS link between routers
  – the circuit ids are leaked to a network attacker
• Predictable malleability might be a useful notion for simulation-based proofs
  – e.g., for protocols that use efficient but malleable stream ciphers
Thank you!

Questions?