Anonymous communication with on-line and off-line onion encoding

Marek Klonowski, Mirosław Kutyłowski, Filip Zagórski

Wrocław University of Technology,
SOFSEM’2005

Partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS)
Privacy in Communication Systems

- messages can be kept secret
- reliable authentication
- how to hide that two parties are communicating??
Need of Anonymity in Communication

- a health insurance company discovers that an applicant has sought information on specific heart diseases – his application get rejected!

- buying a product – the seller knows where I have checked the prices.
  – the game becomes unfair!
Design Goals

- provable security
Design Goals

- provable security
- scalability
- layered approach consistent with communication systems architecture
Design Goals

- provable security
- scalability
- layered approach consistent with communication systems architecture
- adaptiveness to network load
- the end-user machine has limited knowledge of the network
Design Goals

- provable security
- scalability
- layered approach consistent with communication systems architecture
- adaptiveness to network load
- the end-user machine has limited knowledge of the network
- resistance against dynamic attacks (not only observing the network but also inserting/deleting messages)
Naive or Local Network Solutions

- **all-to-all**: send the encrypted message to all participants, communication overhead!
Naive or Local Network Solutions

- **all-to-all**: send the encrypted message to all participants, communication overhead!
- **token ring**: encoded messages go around the ring communication delay!
Onion Encoding

\[ m \]
Onion Encoding
Onion Encoding
Onion Encoding
Onion Encoding
Onion Encoding

$Y$
Onion Encoding
Onion decryption
Anonymity
Existing Solutions
Existing problems

Onion decryption

\[ m \times Y \]
Onion decryption
Onion decryption
Onion decryption
Onion decryption
Onion decryption

\[ m \]
Route of an Onion

single onion

A

B
Route of an Onion

single onion

A

B

Marek Klonowski, Mirosław Kutylowski, Filip Zagórski

Anonymous communication with on-line and off-line onion encoding
Route of an Onion

single onion

A

B
Route of an Onion

- single onion

Diagram showing the route of an onion from A to B.
Route of an Onion

single onion

A → B
Route of an Onion

single onion

A

B
Route of an Onion

single onion

A

B
Route of an Onion

single onion

A

B
If $A$ wants to send a message $m$ to server $B$, $A$ chooses at random $\lambda$ intermediate nodes $J_1, \ldots, J_{\lambda}$;
Classical Onions

If $A$ wants to send a message $m$ to server $B$

- $A$ chooses at random $\lambda$ intermediate nodes $J_1, \ldots, J_\lambda$;
- $A$ creates an onion:

$$O := \text{Enc}_B(m)$$
Classical Onions

If $A$ wants to send a message $m$ to server $B$

- $A$ chooses at random $\lambda$ intermediate nodes $J_1, \ldots, J_\lambda$;
- $A$ creates an onion:
  
  $$O : = \text{Enc}_{J_\lambda} (\text{Enc}_B(m), B)$$
Classical Onions

If $A$ wants send a message $m$ to server $B$

- $A$ chooses at random $\lambda$ intermediate nodes $J_1, \ldots, J_\lambda$;
- $A$ creates an onion:

$$O := \text{Enc}_{J_{\lambda-1}}(\text{Enc}_{J_\lambda}(\text{Enc}_B(m), B), J_\lambda)$$
Classical Onions

If $A$ wants send a message $m$ to server $B$

- $A$ chooses at random $\lambda$ intermediate nodes $J_1, \ldots, J_\lambda$;
- $A$ creates an onion:
  
  $$O := \text{Enc}_{J_1}(\ldots(\text{Enc}_{J_{\lambda-1}}(\text{Enc}_{J_\lambda}(\text{Enc}_B(m), B), J_\lambda), J_{\lambda-1}) \ldots, J_2) .$$
Processing an Onion

If $A$ wants to send a message $m$ encrypted as $O$ to server $B$
- $A$ sends onion $O$ to $J_1$
Processing an Onion

If $A$ wants to send a message $m$ encrypted as $O$ to server $B$

- $A$ sends onion $O$ to $J_1$
- $J_1$ decrypts $O$ and obtains some $(O', J_2)$
Processing an Onion

If $A$ wants to send a message $m$ encrypted as $O$ to server $B$

- $A$ sends onion $O$ to $J_1$
- $J_1$ decrypts $O$ and obtains some $(O', J_2)$
- $J_1$ sends $O'$ to $J_2$
Processing an Onion

If $A$ wants to send a message $m$ encrypted as $O$ to server $B$

- $A$ sends onion $O$ to $J_1$
- $J_1$ decrypts $O$ and obtains some $(O', J_2)$
- $J_1$ sends $O'$ to $J_2$
- $J_2$ decrypts ..
- $J_2$ sends .. to $J_3$
Processing an Onion

If $A$ wants to send a message $m$ encrypted as $O$ to server $B$

- $A$ sends onion $O$ to $J_1$
- $J_1$ decrypts $O$ and obtains some $(O', J_2)$
- $J_1$ sends $O'$ to $J_2$
- $J_2$ decrypts ..
- $J_2$ sends .. to $J_3$
- ...

Onions at Work
Onions at Work

many onions
Onions at Work

many onions

Marek Klonowski, Mirosław Kutyłowski, Filip Zagórski
Anonymous communication with on-line and off-line onion encoding
Onions at Work

many onions
Onions at Work

many onions
Onions at Work

many onions
Onions at Work

many onions
Onions at Work

many onions

Marek Klonowski, Miroslaw Kutylowski, Filip Zagórski
Anonymous communication with on-line and off-line onion encoding
Onions at Work

many onions

destination of the message starting at A?
Disadvantages – Repetitive Attack

an adversary re-sends the same onion

Diagram: Nodes connected in a chain, indicating the sequence of messages in an onion routing protocol.
Disadvantages – Repetitive Attack

an adversary re-sends the same onion
Disadvantages – Repetitive Attack

an adversary re-sends the same onion

![Diagram of a network showing repetitive attacks]
Problem Solution: Universal Re-Encryption

technique due to P. Golle, M. Jakobsson, A. Juels, P. Syverson

- ciphertext obtained with a public key of recipient Alice but everybody can re-code it without knowing the public key of Alice or her identity
- any connection between a ciphertext before and after re-coding undetectable by a third party
- perfect tool for an anonymous re-mailer, ...
URE setup

- $q$ - prime, $G$ - a group of rank $q$ with hard discrete logarithm problem
- $g$ - generator of $G$,
- $x < q$ - private key of Alice
- $y = g^x$ - public key of Alice
URE Ciphertexts

**Encryption:**

\(k_0, k_1 - \text{random}\)

A ciphertext of \(m\):

\[
(\alpha_0, \beta_0; \alpha_1, \beta_1) := (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})
\]
URE Ciphertexts

Encryption:
$k_0, k_1$ - random

A ciphertext of $m$:

$$(\alpha_0, \beta_0; \alpha_1, \beta_1) := (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})$$

Re-encryption:
$k_0', k_1' - random$

The message after re-encryption:

$$
\left(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'}\right) = 
\left(m \cdot y^{k_0 + k_1 \cdot k_0'}, g^{k_0 + k_1 \cdot k_0'}; y^{k_1 \cdot k_1'}, g^{k_1 \cdot k_1'}\right)
$$
Decryption

\((\alpha_0, \beta_0; \alpha_1, \beta_1)\)

Like for ElGamal:

\[
m := \frac{\alpha_0}{\beta_0^x}
\]

\[
m' := \frac{\alpha_1}{\beta_1^x}
\]

A message \(m\) is accepted \(\Leftrightarrow m' = 1\)
URE-Onions

- an URE-onion consists of $\lambda$ blocks
- a block = URE ciphertext
URE-Onions

- an URE-onion consists of $\lambda$ blocks
- a block = URE ciphertext
- encoded plaintexts: $J_2, J_3, \ldots, J_\lambda, m$

- advantage: each block can be re-encrypted while processing at a server
  repetitions get undetected!
URE-Onions - Partial Decryption

Goal: enforce processing along the path

- \( y_1, \ldots, y_\lambda \) = public keys of \( J_1, \ldots, J_\lambda \)
- ciphertext of \( J_i \) encoded with the public key \( y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1} \):

\[
(J_i \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^k, g^k, (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^{k'}, g^{k'})
\]
URE-Onions - Partial Decryption

Goal: enforce processing along the path

- $y_1, ..., y_\lambda = \text{public keys of } J_1, \ldots, J_\lambda$
- ciphertext of $J_i$ encoded with the public key $y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1}$:

\[
(J_i \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^k, g^k, (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^{k'}, g^{k'})
\]

- partial decryption of $(a, b, c, d)$ by $J_1$:

\[
a := a/b^{x_1}, \quad c := c/d^{x_1}
\]
URE-Onions - Partial Decryption

Goal: enforce processing along the path

- $y_1, \ldots, y_\lambda$ = public keys of $J_1, \ldots, J_\lambda$
- ciphertext of $J_i$ – with the public key $y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1}$:
  \[
  (J_i \cdot (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^k, g^k, (y_1 \cdot y_2 \cdot \ldots \cdot y_{i-1})^{k'}, g^{k'})
  \]

- partial decryption of $(a, b, c, d)$ by $J_1$:
  \[
  a := a/b^{x_1}, \quad c := c/d^{x_1}
  \]

Result:

\[
(J_i \cdot (y_2 \cdot \ldots \cdot y_{i-1})^k, g^k, (y_2 \cdot \ldots \cdot y_{i-1})^{k'}, g^{k'})
\]
Processing an Onion

- partial decryption of all blocks
  ⇒ the next hop address $J_i$ or $m$ is retrieved
Processing an Onion

- partial decryption of all blocks
  ⇒ the next hop address $J_i$ or $m$ is retrieved
- re-encryption of all blocks
Processing an Onion

- partial decryption of all blocks
  ⇒ the next hop address $J_i$ or $m$ is retrieved
- re-encryption of all blocks
- random permutation of all blocks
Processing an Onion

- partial decryption of all blocks
  \[ \Rightarrow \text{the next hop address } J_i \text{ or } m \text{ is retrieved} \]
- re-encryption of all blocks
- random permutation of all blocks
- delivery to \( J_i \) or to the final destination
Further Possibilities: Inserting a Ciphertext

Empty container:

\[(a, b, c, d) = (1 \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})\]

Inserting \(m\):

\[a := a \cdot m\]

Result:

\[(a, b, c, d) = (m \cdot y^{k_0}, g^{k_0}; y^{k_1}, g^{k_1})\]
Navigators

Navigators $\equiv$ „empty onions”

$Nav[J_1, \ldots, J_\lambda] = O_{y_1, \ldots, y_\lambda}(-)$
Online Merge Onions

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

\[
\begin{align*}
A & \quad S_1 \\
B &
\end{align*}
\]
Online Merge Onions

A

S₁

B

S₂

Anonymous communication with on-line and off-line onion encoding

Marek Klonowski, Mirosław Kutylowski, Filip Zagórski
Online Merge Onions

\[ A \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow B \]

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

A

S₁

S₂

B

S₃
Online Merge Onions

\[ A \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow B \]
Online Merge Onions

A

S_1

S_2

S_3

B

Anonymous communication with on-line and off-line onion encoding

Marek Klonowski, Mirosław Kutylowski, Filip Zagórski
Online Merge Onions

A

S1

S2

S3

B
Online Merge Onions

Marek Klonowski, Mirosław Kutyłowski, Filip Zagórski
Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

\[ A \xrightarrow{} S_1 \xrightarrow{} S_2 \xrightarrow{} B \]

\[ A \xrightarrow{} S_1 \xrightarrow{} S_3 \]

Marek Klonowski, Mirosław Kutylowski, Filip Zagórski

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

A

S₁

S₂

S₃

B
Online Merge Onions

\[ A \quad S_1 \quad S_2 \quad S_3 \quad B \]

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

\[ A \] \rightarrow S_1 \rightarrow S_2 \rightarrow \ldots \rightarrow S_3 \rightarrow B \]
Online Merge Onions
Online Merge Onions

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

\[ A \xrightarrow{S_1} S_2 \xrightarrow{S_3} B \]
Online Merge Onions

- Universal Re-encryption
  - URE-Onions
  - Online Merge Onions

A
S1

B
S2

S3

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

\[ A \rightarrow S_1 \rightarrow S_2 \rightarrow B \]

\[ S_1 \rightarrow S_3 \]

Marek Klonowski, Mirosław Kutyłowski, Filip Zagórski

Anonymous communication with on-line and off-line onion encoding
Online Merge Onions

A \rightarrow S_1 \rightarrow S_2 \rightarrow B

S_1 \rightarrow S_3
Online Merge Onions
Online Merge Onions

A

B

S1

S2

S3
Online Merge Onions - creation

A has a message $m$ for $B$. Then $A$:

- chooses at random $k$ servers $S_1, \ldots, S_k$
Online Merge Onions - creation

A has a message $m$ for $B$. Then $A$:

- chooses at random $k$ servers $S_1, \ldots, S_k$
- creates a navigator $N = \text{Nav}[S_1, \ldots, S_k]$
Online Merge Onions - creation

A has a message $m$ for $B$. Then $A$:

- chooses at random $k$ servers $S_1, \ldots, S_k$
- creates a navigator $N = Nav[S_1, \ldots, S_k]$
- inserts message „to B” into $N$
Online Merge Onions - creation

A has a message $m$ for $B$. Then $A$:

- chooses at random $k$ servers $S_1, \ldots, S_k$
- creates a navigator $N = Nav[S_1, \ldots, S_k]$
- inserts message „to $B$” into $N$
- creates a ciphertext $URE_{y_B}(m)$ with $y_B$, decryption key of $B$
Online Merge Onions - creation

A has a message $m$ for $B$. Then $A$:

- chooses at random $k$ servers $S_1, \ldots, S_k$
- creates a navigator $N = Nav[S_1, \ldots, S_k]$
- inserts message „to B” into $N$
- creates a ciphertext $URE_{y_B}(m)$ with $y_B$, decryption key of $B$
- sends to $S_1$:

\[
Nav[S_1, S_k](to \ B) \ , \ URE_{y_B}(m)
\]
Online Merge Onions – processing

A message obtained by a server on a path of $m$ consists of:

- $Nav[J_i, J_m](toS_j)$ – “local navigator” chosen online
- $URE(Nav[S_j, S_k](toB))$ – ciphertext of the remaining part of the “global navigator”
- $URE_{yB}(m)$
Online Merge Onions – processing

A message obtained by a server on a path of $m$ consists of:

- $Nav[J_i, J_m](toS_j)$ – “local navigator” chosen online
- $URE(Nav[S_j, S_k](toB))$ – ciphertext of the remaining part of the “global navigator”
- $URE_{yB}(m)$

The $i$th server from the list $J_1, \ldots, J_l$ proceeds:

- partial decryption of navigators
- re-encryption
- sending according to the “internal navigator”
Online Merge Onions – processing

A message obtained by a server on a path of $m$ consists of:

- $Nav[J_i, J_m](toS_j)$ – “local navigator” chosen online
- $URE(Nav[S_j, S_k](toB))$ – ciphertext of the remaining part of the “global navigator”
- $URE_{y_B}(m)$

The $i$th server from the list $S_1, ..., S_k$ proceeds:

- retrieves $Nav[S_{i+1}, S_k]$ with its private key
- chooses a local navigator $M[J_1, ..., J_l]$ and inserts the message “to $S_{i+1}$”
- URE-encrypts $Nav[S_{i+1}, S_k]$ for this path
- sends to $J_1$
Online Merge Onions - repetitive attack

repetitive attack?
Online Merge Onions - repetitive attack

repetitive attack?
Online Merge Onions - repetitive attack

repetitive attack?
Online Merge Onions - repetitive attack

repetitive attack?
Online Merge Onions - repetitive attack

A

S₁

S₂

B

S₃

repeatitive attack?
Further Advantages

- if different users compose paths from different sets of servers (in the classical approach), then breaking anonymity is possible.
- online onions – the users compose navigators from a fixed stable set of servers.
Further Advantages

- if different users compose paths from different sets of servers (in the classical approach), then breaking anonymity is possible.

Online onions – the users compose navigators from a fixed stable set of servers.

- enforcing "vertex mixing" helps to reduce the paths lengths without loosing provable privacy.
Further Advantages

- if different users compose paths from **different sets of servers** (in the classical approach), then **breaking anonymity is possible**
  - online onions – the users compose navigators from a fixed stable set of servers
- enforcing „**vertex mixing”** helps to **reduce the paths lengths without loosing provable privacy**
- **adaptiveness**: high traffic $\Rightarrow$ the paths can be shorter
  - reduction of communication overhead
Further Advantages

- if different users compose paths from **different sets of servers** (in the classical approach), then breaking anonymity is possible
- online onions – the users compose navigators from a fixed stable set of servers
- enforcing „**vertex mixing**“ helps to reduce the paths lengths without loosing provable privacy
- **adaptiveness**: high traffic $\Rightarrow$ the paths can be shorter reduction of communication overhead
- **layered architecture**
- **onions can be prepared in advance**
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message size</td>
<td>$S = O(\lambda +</td>
<td>m</td>
</tr>
<tr>
<td>Traffic change</td>
<td>– decrease</td>
<td>yes</td>
</tr>
<tr>
<td>Required knowledge of network topology</td>
<td>limited</td>
<td>yes</td>
</tr>
<tr>
<td>Adaptiveness</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>$S = O(\lambda +</td>
<td>m</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>( S = O(\lambda +</td>
<td>m</td>
</tr>
<tr>
<td>preprocessing possible</td>
<td>no</td>
<td>partially</td>
</tr>
</tbody>
</table>

\( |m| \) is the number of messages, and \( \lambda \) represents some constant.
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>$S=O(\lambda+</td>
<td>m</td>
</tr>
<tr>
<td>preprocessing possible</td>
<td>no</td>
<td>partially</td>
</tr>
</tbody>
</table>

- **Classical Onions**: The message size is given by $S=O(\lambda+|m|)$. Preprocessing is not possible.
- **Online Merge Onions**: The message size is approximately 4 times the size of the classical onions. Partial preprocessing is possible.

- **Comparison of Attacks**:
  - Message tracing: Easy for classical onions, harder for online merge onions.
  - Repetitive attack: Easy for classical onions, harder for online merge onions.
  - Traffic change: Requires knowledge of network topology. Limited adaptiveness for classical onions, full adaptiveness for online merge onions.
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>message size</strong></td>
<td>$S=O(\lambda +</td>
<td>m</td>
</tr>
<tr>
<td><strong>preprocessing possible</strong></td>
<td>no</td>
<td>partially</td>
</tr>
<tr>
<td><strong>messages tracing</strong></td>
<td>easy</td>
<td>hard</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>$S=O(\lambda+</td>
<td>m</td>
</tr>
<tr>
<td>preprocessing possible</td>
<td>no</td>
<td>partially</td>
</tr>
<tr>
<td>messages tracing*</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>repetitive attack**</td>
<td>easy</td>
<td>harder</td>
</tr>
</tbody>
</table>

* Traffic change required knowledge of network topology. ** Full adaptiveness, limited adaptiveness. 

Anonymous communication with on-line and off-line onion encoding.
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>$S = O(\lambda +</td>
<td>m</td>
</tr>
<tr>
<td>preprocessing possible</td>
<td>no</td>
<td>partially</td>
</tr>
<tr>
<td>messages tracing*</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>repetitive attack**</td>
<td>easy</td>
<td>harder</td>
</tr>
<tr>
<td>traffic change</td>
<td>—</td>
<td>decrease</td>
</tr>
</tbody>
</table>
Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>$S = O(\lambda +</td>
<td>m</td>
</tr>
<tr>
<td>preprocessing possible</td>
<td>no</td>
<td>partially</td>
</tr>
<tr>
<td>messages tracing*</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>repetitive attack**</td>
<td>easy</td>
<td>harder</td>
</tr>
<tr>
<td>traffic change</td>
<td>—</td>
<td>decrease</td>
</tr>
<tr>
<td>required knowledge of network topology</td>
<td>full</td>
<td>limited</td>
</tr>
</tbody>
</table>
### Comparison

<table>
<thead>
<tr>
<th></th>
<th>Classical Onions</th>
<th>Online Merge Onions</th>
</tr>
</thead>
<tbody>
<tr>
<td>message size</td>
<td>$S = O(\lambda +</td>
<td>m</td>
</tr>
<tr>
<td>preprocessing possible</td>
<td>no</td>
<td>partially</td>
</tr>
<tr>
<td>messages tracing*</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>repetitive attack**</td>
<td>easy</td>
<td>harder</td>
</tr>
<tr>
<td>traffic change</td>
<td>—</td>
<td>decrease</td>
</tr>
<tr>
<td>required knowledge of network topology</td>
<td>full</td>
<td>limited</td>
</tr>
<tr>
<td>traffic adaptiveness</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Thank you for attention!