ModOnions
Onion Routing
Modified Onion Routing
Attacks on ModOnions
Detour Attack
Tagging Attack
Defence
Core Idea
Improved Construction
Routing
Security

Repelling Detour Attack against Onions with Re-Encryption

Marek Klonowski Mirosław Kutyłowski **Anna Lauks**

Wrocław University of Technology

ACNS 2008

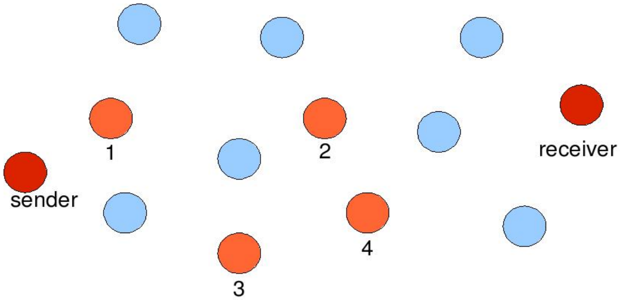
Outline

1. ModOnions
   - Onion Routing
   - Modified Onion Routing
2. Attacks on ModOnions
   - Detour Attack
   - Tagging Attack
3. Defence
   - Core Idea
   - Improved Construction
   - Routing
4. Security

Outline

1. ModOnions
   - Onion Routing
   - Modified Onion Routing
2. Attacks on ModOnions
   - Detour Attack
   - Tagging Attack
3. Defence
   - Core Idea
   - Improved Construction
   - Routing
4. Security

Onion Routing



$$E_{pk_1}(\text{"2"}, E_{pk_2}(\text{"3"}, E_{pk_3}(\text{"4"}, E_{pk_4}(\text{"receiver"}, E_{pk_r}(m)))))$$

Onion Routing

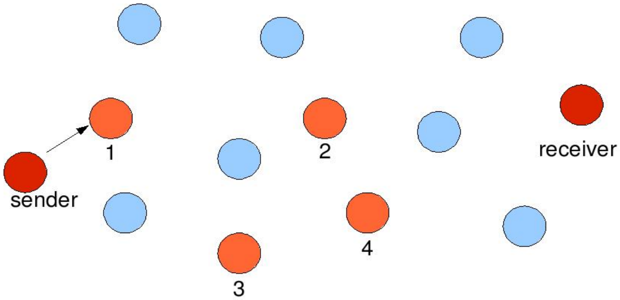
Anna Lauks

ModOnions
- Onion Routing
- Modified Onion Routing

Attacks on ModOnions
- Detour Attack
- Tagging Attack

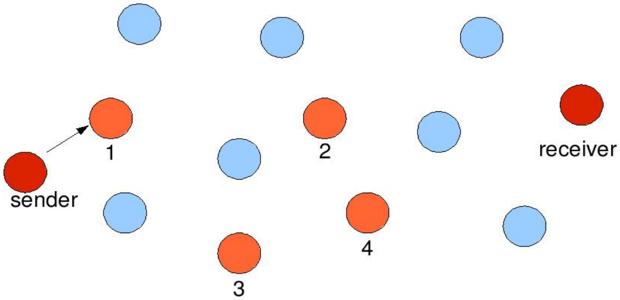
Defence
- Core Idea
- Improved Construction
- Routing

Security



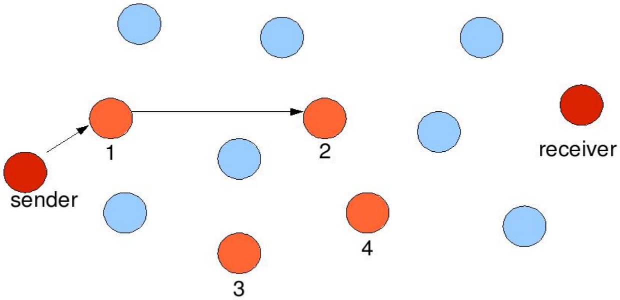
$$E_{pk_1}(\text{"2"}, E_{pk_2}(\text{"3"}, E_{pk_3}(\text{"4"}, E_{pk_4}(\text{"receiver"}, E_{pk_r}(m)))))$$

Onion Routing



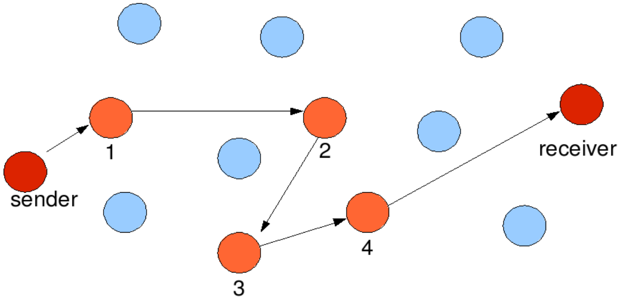
$$E_{pk_2}(\text{"3"}, E_{pk_3}(\text{"4"}, E_{pk_4}(\text{"receiver"}, E_{pk_r}(m))))$$

Onion Routing



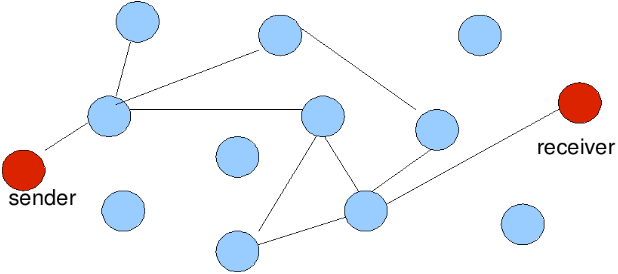
$$E_{pk_2}(\text{"3"}, E_{pk_3}(\text{"4"}, E_{pk_4}(\text{"receiver"}, E_{pk_r}(m))))$$

Onion Routing



Receiver decrypts and gets message m

Onion Routing - Anonymity



- 2 Onions entering and leaving the same node are indistinguishable – conflict
- problem: replay attack

ModOnions Protocol – [1]

Basic Properties

- utilizes extended version of Universal Re-encryption (from [2])
- each Onion consists of λ ciphertexts (called ,,blocks")
- additional phase while routing – re-encryption of all blocks of the Onion – immunity against replay attack

[1] M. Gomułkiewicz, M. Kutyłowski: "Onions Based on Universal Re-encryption – Anonymous Communication Against Repetitive Attack"

[2] P. Golle, M. Jakobsson, A. Juels, P.F. Syverson: "Universal Re-encryption for Mixnets"

ModOnions Protocol – Building Blocks

Extended Version of Universal Re-encryption

Keys: $x_i, \quad y_i = g^{x_i}$ – private and public key of the ith server

Encryption: $E_{x_1+x_2+\dots+x_\lambda}(m) = (\alpha_0, \ \beta_0; \ \alpha_1, \ \beta_1) :=$
$:= (m \cdot (y_1 y_2 \dots y_\lambda)^{k_0}, \ g^{k_0}; \ (y_1 y_2 \dots y_\lambda)^{k_1}, \ g^{k_1}),$
for some random values k_0 and k_1

ModOnions Protocol – Building Blocks

Extended Version of Universal Re-encryption

Keys: $x_i, \quad y_i = g^{x_i}$ – private and public key of the ith server

Encryption: $E_{x_1+x_2+\dots+x_\lambda}(m) = (\alpha_0,\ \beta_0;\ \alpha_1,\ \beta_1) :=$
$:= (m \cdot (y_1 y_2 \dots y_\lambda)^{k_0},\ g^{k_0};\ (y_1 y_2 \dots y_\lambda)^{k_1},\ g^{k_1})$,
for some random values k_0 and k_1

Re-encryption: $(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'})$ for some random values k_0' and k_1'

ModOnions Protocol – Building Blocks

Extended Version of Universal Re-encryption

Keys: $x_i, \quad y_i = g^{x_i}$ – private and public key of the ith server

Encryption: $E_{x_1+x_2+\dots+x_\lambda}(m) = (\alpha_0,\ \beta_0;\ \alpha_1,\ \beta_1) :=$
$:= (m \cdot (y_1 y_2 \dots y_\lambda)^{k_0},\ g^{k_0};\ (y_1 y_2 \dots y_\lambda)^{k_1},\ g^{k_1})$,
for some random values k_0 and k_1

Re-encryption: $(\alpha_0 \cdot \alpha_1^{k_0'}, \beta_0 \cdot \beta_1^{k_0'}; \alpha_1^{k_1'}, \beta_1^{k_1'})$ for some random values k_0' and k_1'

Partial decryption: ith server can compute:

$$E_{x_1+x_2+\dots+x_{i-1}+x_{i+1}+\dots+x_\lambda}(m) = \left(\frac{\alpha_0}{\beta_0^{x_i}}, \beta_0; \frac{\alpha_1}{\beta_1^{x_i}}, \beta_1\right)$$

ModOnions Protocol

The Goal – to send a message m to server s_λ

Construction of ModOnion – $\mathcal{O}$

- intermediate servers $s_1, s_2, \ldots, s_{\lambda-1}$ are chosen at random
- the i^{th} block of $\mathcal{O}$ (for $1 \leq i < \lambda$) is a ciphertext: $E_{x_{s_1}+\cdots+x_{s_i}}(\text{send to } s_{i+1})$
- the last block of $\mathcal{O}$ has the form: $E_{x_{s_1}+\cdots+x_{s_\lambda}}(m)$
- all blocks are permuted at random and $\mathcal{O}$ is sent to s_1

ModOnions Protocol

Routing of ModOnion – $\mathcal{O}$ by server s_i

- All blocks of $\mathcal{O}$ are:
  1. partially decrypted – only one block should contain plaintext – address of the next server on the path s_{i+1} (it is replaced with random strings)
  2. re-encrypted
  3. permuted at random
- ModOnion $\mathcal{O}$ is sent to s_{i+1}

ModOnions Protocol

Routing of ModOnion – $\mathcal{O}$ by server s_i

- All blocks of $\mathcal{O}$ are:
  1. partially decrypted – only one block should contain plaintext – address of the next server on the path s_{i+1} (it is replaced with random strings)
  2. re-encrypted
  3. permuted at random
- ModOnion $\mathcal{O}$ is sent to s_{i+1}

If any misbehaviour is detected (i.e. **none** or **more then one** decrypted block represents the name of the server) an **investigation procedure** is executed.

Outline

1 ModOnions
- Onion Routing
- Modified Onion Routing

2 Attacks on ModOnions
- Detour Attack
- Tagging Attack

3 Defence
- Core Idea
- Improved Construction
- Routing

4 Security

Attacks on ModOnions from [1]

Observations

Given $E_x(m)$:

1. It is easy to create $E_{x+x'}(m)$ for an arbitrary value x' – one can add an additional cryptographic layer to an arbitrary block of ModOnion

[1] G. Danezis: "Breaking Four Mix-Related Schemes Based on Universal Re-encryption"

Attacks on ModOnions from [1]

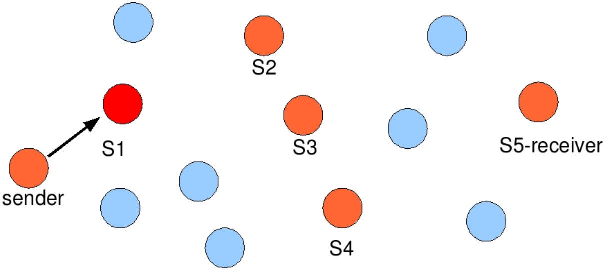
Observations

Given $E_x(m)$:

1. It is easy to create $E_{x+x'}(m)$ for an arbitrary value x' – one can add an additional cryptographic layer to an arbitrary block of ModOnion
2. It is easy to obtain $E_x(m')$ – one can obtain a ciphertext of an arbitrary message m' for the same secret key

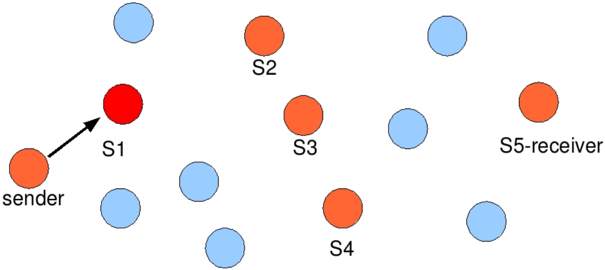
[1] G. Danezis: "Breaking Four Mix-Related Schemes Based on Universal Re-encryption"

Description of the Detour Attack



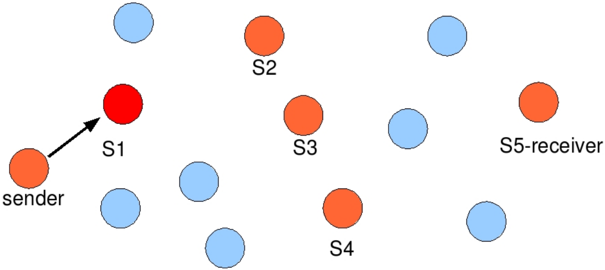
$E_{x_{s_1}}(s_2)$,
$E_{x_{s_1}+x_{s_2}}(s_3)$,
$E_{x_{s_1}+x_{s_2}+x_{s_3}}(s_4)$,
$E_{x_{s_1}+x_{s_2}+x_{s_3}+x_{s_4}}(s_5)$,
$E_{x_{s_1}+x_{s_2}+x_{s_3}+x_{s_4}+x_{s_5}}(m)$

Description of the Detour Attack



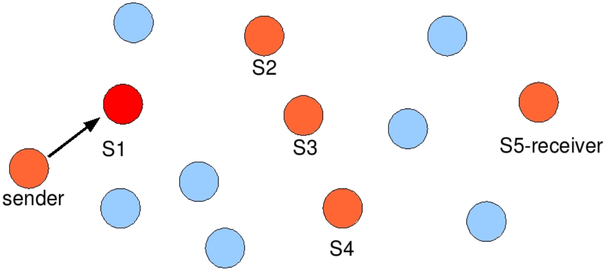
$$E_0(\mathbf{s_2}),$$
$$E_{x_{s_2}}(s_3),$$
$$E_{x_{s_2}+x_{s_3}}(s_4),$$
$$E_{x_{s_2}+x_{s_3}+x_{s_4}}(s_5),$$
$$E_{x_{s_2}+x_{s_3}+x_{s_4}+x_{s_5}}(m)$$

Description of the Detour Attack



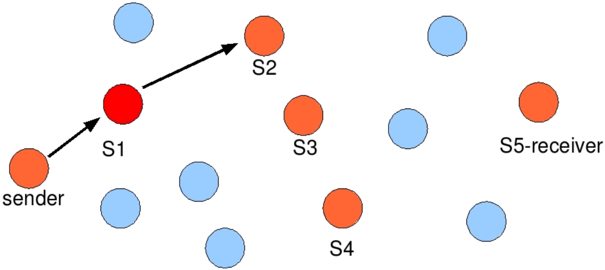
$$E_0(\mathbf{s_2}),$$
$$E_{x_{s_2}+x'}(s_3),$$
$$E_{x_{s_2}+x_{s_3}+x'}(s_4),$$
$$E_{x_{s_2}+x_{s_3}+x_{s_4}+x'}(s_5),$$
$$E_{x_{s_2}+x_{s_3}+x_{s_4}+x_{s_5}+x'}(m)$$

Description of the Detour Attack



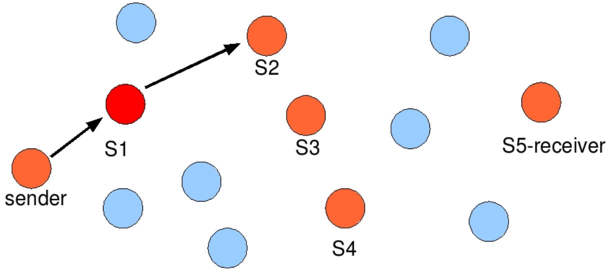
$E_{x_{s_2}}(s_1)$, $\leftarrow$ redirectional block
$E_{x_{s_2}+x'}(s_3)$,
$E_{x_{s_2}+x_{s_3}+x'}(s_4)$,
$E_{x_{s_2}+x_{s_3}+x_{s_4}+x'}(s_5)$,
$E_{x_{s_2}+x_{s_3}+x_{s_4}+x_{s_5}+x'}(m)$

Description of the Detour Attack



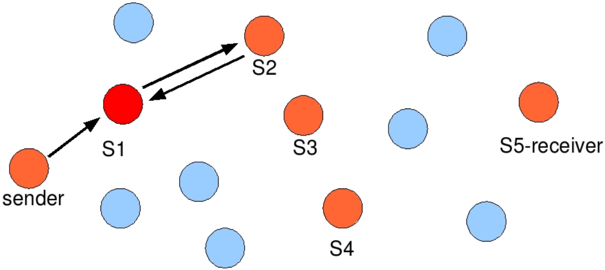
$$E_{x_{s_2}}(s_1),$$
$$E_{x_{s_2}+x'}(s_3),$$
$$E_{x_{s_2}+x_{s_3}+x'}(s_4),$$
$$E_{x_{s_2}+x_{s_3}+x_{s_4}+x'}(s_5),$$
$$E_{x_{s_2}+x_{s_3}+x_{s_4}+x_{s_5}+x'}(m)$$

Description of the Detour Attack



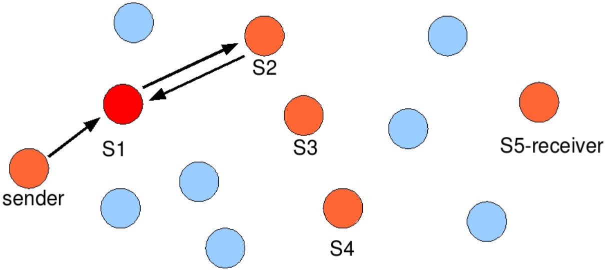
$E_0(s_1)$,
$E_{x'}(s_3)$,
$E_{x_{s_3}+x'}(s_4)$,
$E_{x_{s_3}+x_{s_4}+x'}(s_5)$,
$E_{x_{s_3}+x_{s_4}+x_{s_5}+x'}(m)$

Description of the Detour Attack



random strings,
$E_{x'}(s_3)$,
$E_{x_{s_3}+x'}(s_4)$,
$E_{x_{s_3}+x_{s_4}+x'}(s_5)$,
$E_{x_{s_3}+x_{s_4}+x_{s_5}+x'}(m)$

Description of the Detour Attack



random strings,

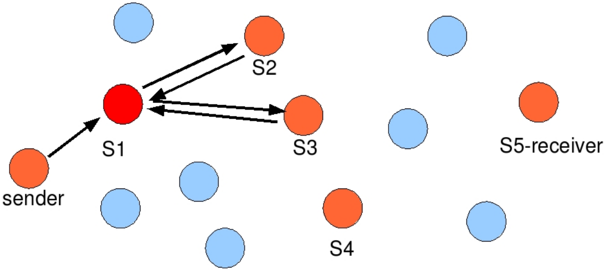
$E_0(s_3)$, $\leftarrow$ s_1 gets the knowledge about s_3!

$E_{x_{s_3}}(s_4)$,

$E_{x_{s_3}+x_{s_4}}(s_5)$,

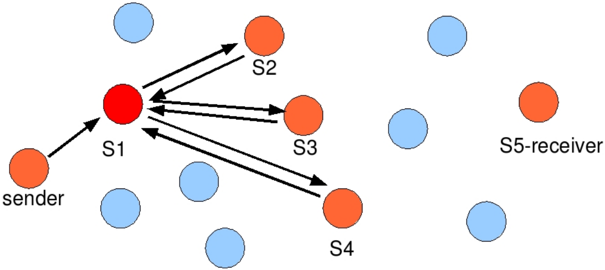
$E_{x_{s_3}+x_{s_4}+x_{s_5}}(m)$

Description of the Detour Attack



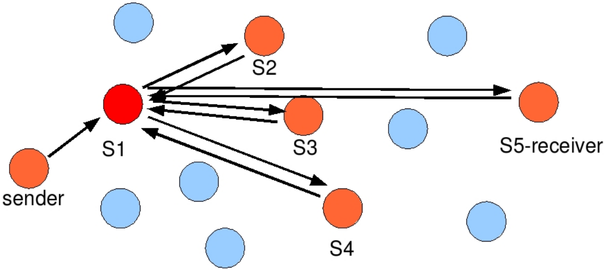
random strings,
random strings,
$E_0(s_4)$, $\leftarrow$ s_1 gets the knowledge about s_4!
$E_{x_{s_4}}(s_5)$,
$E_{x_{s_4}+x_{s_5}}(m)$

Description of the Detour Attack



random strings,
random strings,
random strings,
$E_0(s_5)$, $\leftarrow s_1$ gets the knowledge about s_5!
$E_{x_{s_4}+x_{s_5}}(m)$

Description of the Detour Attack



random strings,
random strings,
random strings,
random strings,
$E_0(m)$ $\leftarrow$ s_1 gets the knowledge about the message m!

Tagging Attack

Idea

1. Corrupted server guesses that the next say 3 hops of ModOnion are s_A, s_B, s_C
2. He marks ModOnion by replacing a **random** block by:

$$E_{x_{s_A}+x_{s_B}+x_{s_C}}(\text{TAG})$$

3. If the path is exactly as he has thought the TAG will be visible

Outline

1 ModOnions
- Onion Routing
- Modified Onion Routing

2 Attacks on ModOnions
- Detour Attack
- Tagging Attack

3 Defence
- Core Idea
- Improved Construction
- Routing

4 Security

Modifications of the ModOnion's Protocol

Core Idea

Each server s has two pairs of keys:

- **transport keys**: private x_s and public $y_s = g^{x_s}$
  - used for transporting blocks through intermediate servers
- **destination keys**: private $x_s^\star$ and public $y_s^\star = g^{x_s^\star}$
  - used for encrypting and decrypting messages and routing addresses for their recipients

Modifications of the ModOnion's Protocol

New Construction of ModOnion $\mathcal{O}$

- the 1^{st} block of $\mathcal{O}$ has the form:
  $E_{x^{\star}_{s_1}}(\text{send to } s_2)$
- the i^{th} block of $\mathcal{O}$ (for $2 \leq i \leq \lambda - 1$) is a ciphertext:
  $E_{x_{s_1}+\dots+x_{s_{i-1}}+x^{\star}_{s_i}}(\text{send to } s_{i+1})$
- the last block of $\mathcal{O}$ has the form:
  $E_{x_{s_1}+\dots+x_{s_{\lambda-1}}+x^{\star}_{s_\lambda}}(m, t)$, where t is the current time
- all blocks are permuted at random and $\mathcal{O}$ is sent to s_1

Each **destination** key is used only **once**!

Modifications of the ModOnion's Protocol

New Routing of ModOnion $\mathcal{O}$

Server s_i:

1. Copies all blocks of $\mathcal{O}$
2. Decrypts all blocks with his private **destination** key
   - one should contain the name of the next server
3.
   - if all blocks are meaningless strings $\rightarrow$ the investigation procedure
   - else server s_i decrypts all copies of blocks (except the one with the address) with the private **transportation** key
4. Replaces the block containing s_{i+1} by a random one
5. Permutes all blocks
6. Sends $\mathcal{O}$ to s_{i+1}

Outline

1 ModOnions
- Onion Routing
- Modified Onion Routing

2 Attacks on ModOnions
- Detour Attack
- Tagging Attack

3 Defence
- Core Idea
- Improved Construction
- Routing

4 Security

Immunity against Detour Attack

Detour Attack

If s_i wants to find the s_{i+2} he should:

1. Add the redirectional block $E_{x_{s^\star_{i+1}}}$
   - to enforce server s_{i+1} to send the ModOnion back
2. Add the additional encryption layer (with some key x') to other blocks

Immunity against Detour Attack

Why does it fail?

The attack is succesful $\iff$ s_{i+1} will use his **destination** key to remove the encryption layer from the block that encodes address of s_{i+2} but:

- if server s_{i+1} honest he will use his destination key to decrypt only the redirectional block
- the rest of the blocks will be partialy decrypted with the transportation key

General Security Discussion

Adversary Model

- adversary may control a small fraction of them (i.e. $d = \frac{1}{\lambda}$, where λ - the length of the path)
- sender and receiver are honest

Attack Model

The adversary can:

- observe Onions transmitted
- manipulate Onions processed by servers he controls
  - change the routing path
  - manipulating the contents
- inject new Onions

General Security Discussion

Successful Attack

An attack $\mathcal{A}$ succeeds $\iff$

- adversary can get some information about the contents of the Onion
- and probability that corrupted server will be detected is negligible

General Security Discussion

Offline Attacks

We showed that single ModOnion does not betray any knowledge about:

- the message encoded inside it
- the identity of any server from the path

General Security Discussion

Offline Attacks

We showed that single ModOnion does not betray any knowledge about:

- the message encoded inside it
- the identity of any server from the path

Online Attacks

- any change of the original path of the ModOnion does not lead to the successful attack
- using random blocks for tagging does not lead to the successful attack

Thank you for attention!