Cryptanalysis of a fair anonymity for the tor network

Amadou Moctar Kane

KSecurity, BP 47136, Dakar, Senegal. amadou1@gmail.com

April 16, 2015

Abstract

The aim of this paper is to present an attack upon the protocol of Diaz et al. [4], which goal is to introduce a fair anonymity in the Tor network. This attack allows an attacker to impersonate Tor users with the complicity of an exit node.

Keywords : Cryptography, Tor, privacy, anonymity, group signature.

1 Introduction

Designed for the low-latency anonymous communication [7], Tor is now the major tool intended to reduce the mass surveillance. Unfortunately Tor also became a haven for criminals, which complicates the task of its supporters. In order to correct this situation, a protocol designed to establish a responsible Tor was introduced first by [3] (using secret sharing schemes) and later by Diaz et al. [4] (using group signatures and blind signatures).

The use of blind or group signatures in anonymity revocation exists already in many papers. For example, on the internet, Claessens et al. [1] introduced a revocable anonymity based on blind and group signatures. Later Köpsell et al. [2] proposed a revocable anonymity based on a proxy system, threshold group signatures and blind signature.

In the next section, we present the scheme of Diaz et al. [4], and before the conclusion, we will exhibit the weakness of their protocol and the proposed improvement.

2 Presentation of Diaz et al's scheme

In order to endow Tor with fairness capabilities, this scheme works by introducing variations in the way a user negotiates the symmetric keys with the entry and exit nodes. It introduces the group signature, the blind signature, the blind group signature, the commitment and other cryptographic tools during the handshake with the entry and the exit node. In addition, in order to prevent the user to employ one identity for negotiating with the entry node, and a different one with the exit node, the entry node has to blindly sign the message that the user will send to the exit node. The resulting modified handshake schemes are shown below, where U_i denotes any arbitrary user, EN denotes the entry node and EX the exit node. During the handshake with EN, U_i first group-signs g^{x_1} and g^{x_2}, sends g^{x_1} to EN and also requests EN to blindly sign a group signature of g^{x_2}. If all the operations succeed, EN accepts the connection.

2.1 Entry Node Handshake:

U_i: $\sigma_1 \leftarrow GS.Sign(g^{x_1}, mk_i)$
U_i creates a group signature over g^{x_1}, using her member key mk_i.
U_i: $\sigma_2 \leftarrow GS.Sign(g^{x_2}, mk_i)$
U_i creates a group signature over g^{x_2}, using her member key mk_i.
U_i: $com \leftarrow Com(\sigma_2, r_1)$
Denote a commitment com to a message σ_2, where the sender uses uniform random coins r_1;
U_i: $(\beta, \pi) \leftarrow BGS.Blind(com, r_2)$
Blind group signature (BGS) scheme is just like a blind signature in which the signer issues a group signature instead of a conventional signature. Using some random secret value (r_2), the user creates a blinded version (β) of the message (com) to be blindly signed and a proof of correctness π.
U_i: $\phi \leftarrow ProveZK(x, w)$ where $x = (\beta, \pi, \sigma_1)$, $w = (mk_i, r_1, r_2)$ such that: $\sigma_2 \leftarrow GS.Sign(g^{x_2}, mk_i)$, $(\beta, \pi) \leftarrow BGS.Blind(Com(\sigma_2, r_1), r_2)$.
$ProveZK(x, w)$ and $VerifyZK$ refer to creating a non-interactive proof showing that the statement x is in the language (which will be determined by the context) with the witness w, and to verify the statement x based on the proof π.
$U_i \rightarrow EN : g^{x_1}, \sigma_1, \beta, \pi, \phi$.
$EN : VerifyZK(\beta, \pi, \phi, \sigma_1)$.
$EN : GS.Verify(\sigma_1, g^{x_1})$.
Allows the EN with the group key to verify the group signature σ_1.
$EN : \tilde{\beta} \leftarrow BGS.Sign(\beta, sbk)$.
Upon receiving the blinded messages, the EN runs any necessary verification and creates a blinded group signature using its private key (sbk).
$EN : K_1 = g^{x_1 y_1}$.
$EN \leftarrow U_i : g^{y_1}, \tilde{\beta}, H(K_1 | h_{sK_1})$.
$U_i : \sigma_3 \leftarrow BGS.Unblind(\tilde{\beta}, r_2)$.
The user receives the blinded group signature and unblinds it, using the secret value (r_2) generated during the blind process.
$U_i : K_1 = g^{x_1 y_1}$.

2.2 Exit Node Handshake:

$U_i \rightarrow EX : g^{x_2}, \sigma_2, \sigma_3$.
U_i initiates the handshake with EX, she sends the group signature on g^{x_2} (σ_3) that was blindly signed by EN, along with the blind signature itself (σ_2).
$EX : GS.Verify(\sigma_2, g^{x_2})$.
Allows the EX with the group key to verify the group signature σ_2.
$EX : BGS.Verify(\sigma_3, \sigma_2)$.
Allows the EX with the group key to verify the blinded signature of σ_2.
$EX : K_2 = g^{x_2 y_2}$.
$EX \rightarrow Ui : g^{y_2}, H(K_2 | h_{sK_2})$.
$U_i : K_2 = g^{x_2 y_2}$.
If all the verifications succeed, then EX accepts the connection.

3 Attack on the protocol

Let's suppose that the exit node (EX) is a malicious relay. In order to conduct this attack, the EX have to copy the elements of the handshake with honest Tor users and then send the copies to its accomplices who are in the network.
Hence, after the connection of U_i, the EX will copy and send g^{x_2}, σ_2 and y_2 , to its accomplice U_j.
U_j can choose any entry node EN, but it should finish by its accomplice (the exit node), it should also choose x_{11}, r_{11}, r_{22} as it is currently done with x_1, r_1, r_2.
Handshake with the EN:
U_j: $\sigma_1 \leftarrow GS.Sign(g^{x_{11}}, mk_j)$
U_j creates a group signature over $g^{x_{11}}$, using its member key mk_j.
U_j: $com \leftarrow Com(\sigma_2, r_{11})$
Denote a commitment com to a message σ_2, where the sender uses uniform random coins r_{11};
U_j: $(\beta, \pi) \leftarrow BGS.Blind(com, r_{22})$
Blind group signature (BGS) scheme is just like a blind signature in which the signer issues a group signature instead of a conventional signature. Using some random secret value (r_{22}), the user creates a blinded version (β) of the message (com) to be blindly signed and a proof of correctness π.
U_i: $\phi \leftarrow ProveZK(x, w)$ where $x = (\beta, \pi, \sigma_1)$, $w = (mk_j, r_{11}, r_{22})$ such that:
$\sigma_2 \leftarrow GS.Sign(g^{x_2}, mk_i)$, $(\beta, \pi) \leftarrow BGS.Blind(Com(\sigma_2, r_{11}), r_{22})$.
$ProveZK(x, w)$ and $VerifyZK$ refer to creating a non-interactive proof showing that the statement x is in the language (which will be determined by the context) with the witness w, and to verify the statement x based on the proof π.
$U_i \rightarrow EN : g^{x_{11}}, \sigma_1, \beta, \pi, \phi$.
$EN : VerifyZK(\beta, \pi, \phi, \sigma_1)$.
$EN : GS.Verify(\sigma_1, g^{x_{11}})$.
Allows the EN with the group key to verify the group signature σ_1.
$EN : \tilde{\beta} \leftarrow BGS.Sign(\beta, sbk)$.
Upon receiving the blinded messages, the EN runs any necessary verification and creates a blinded group signature using its private key (sbk).
$EN : K_1 = g^{x_{11}y_1}$.
$EN \rightarrow U_i : g^{y_1}, \tilde{\beta}, H(K_1|h_{sK_1})$.
$U_i : \sigma_3 \leftarrow BGS.Unblind(\tilde{\beta}, r_{22})$.
The user receives the blinded group signature and unblinds it, using the secret value (r_{22}) generated during the blind process. The result of this operation is the final signature.
$U_i : K_1 = g^{x_{11}y_1}$.
Handshake with EX :
$U_j \rightarrow EX : g^{x_2}, \sigma_2, \sigma_3$.
When EX recognizes g^{x_2}, it actes as follows
$EX : K_2 = g^{x_2y_2}$.
$EX \rightarrow U_j : g^{y_2}, H(K_2|h_sK_2)$. U_j which already have y_2 can create K_2.
$U_j : K_2 = g^{x_2y_2}$, then EX accepts the connection.
The anonymity revocation:
As proposed in [4], Let's suppose that U_j established a circuit and she is communicating with some server S (external to Tor). Also, let us suppose that eventually, U_j performs some illegitimate action. When

that happens, S denounces this behavior following some predefined method. Specifically, the exit node provides the following information to retrieve U_j's identity:

1. $\{msg\}_K$, where msg is the message received and denounced by S, and K is the symmetric key negotiated between U_i and the exit node.
2. $(K = g^{x_2 y_2}, g^{x_2})$, where g^{x_2} is U_i's share of the handshake and g^{y_2} is the share created by the exit node.
3. σ_2, i.e., a group signature of g^{x_2} issued by U_i.

In order to verify that the received denounce is valid, it is necessary to check that the message received from S, msg, corresponds to the encryption $\{msg\}_K$ received from the exit node. Also, σ_2 must be a valid group signature over g^{x_2} . Finally, the exit node may be required to prove that it knows the discrete logarithm y_2 of $g^{x_2 y_2}$ to the base g^{x_2}. If these checks succeed, then the member with key $mk_i(U_i)$ is considere to be the responsible of msg. Hence, U_i's key can be consequently revoked and U_i can be prosecuted while the responsible of this message is U_j.
Remark: If the user has at least two accomplices on the Tor network (an entry node and an exit node), it can impersonate a Tor user without being member of Tor.

3.1 Improvement

In order to correct this weakness, we propose to keep the handshake of Tor as it is currently done [7], except for the last relay where Alice (the sender) will group-sign and timestamp Bob's IP. She will also establish a secure connection with Bob in order to prevent a man-in-the-middle attack (for example a hash was included in the Tor handshake in order to prevent such attack).
The revocation of the key will be done as follows:
A judge (or an organization responsible for the fight against cybercrime) will contact the last relay to tell the EX that at the time t it sent the offending message to Bob. The organization or the judge would first prove that the offending message comes from the secure connection that is passed through the last relay. The last relay will exhibit the IP of Bob which was signed and timestamped by Alice. If all verifications are successful then the group manager will vote in order to revoke or not Alice's member key. This improvement can be implemented in TAP and in Ntor.

3.2 Example in TAP

The beginning of the Tor authentication protocol is:
Alice $\rightarrow OR1 : Create\, c1$, E(g^{x_1}).
OR1 $\rightarrow Alice : Created\, c1$, g^{y_1}, $H(K1)$.
Alice $\rightarrow OR1 : Relay\, c1$ {Extend, OR2, E(g^{x_2})}.
OR1 $\rightarrow OR2 : Create\, c2$ E(g^{x_2}).
OR2 $\rightarrow OR1 : Created\, c2$, g^{y_2}, $H(K2)$.
OR1 $\rightarrow Alice : Relay\, c1$ {Extended, OR2, E(g^{y_2}, $H(K2)$)}.
The single change, which we will introduce in this protocol is to timestamp and to group-sign the IP of the recipient ($\zeta \leftarrow Timestamp.GS.sign(< website >: 443, mk_i)$) and to add it on the following line of TAP.

Alice $\rightarrow OR1 : Relay\, c1$ {{ Begin <website>:443|ζ}} (the concatenation is designated by |).
Before any other action (TCP handshake, ...), OR2 will verify if the timestamp and the group signatures are correct. If these are correct, then it will continue the protocol as described in the first design of Tor [7].

3.3 A revocable anonymity in the hidden services

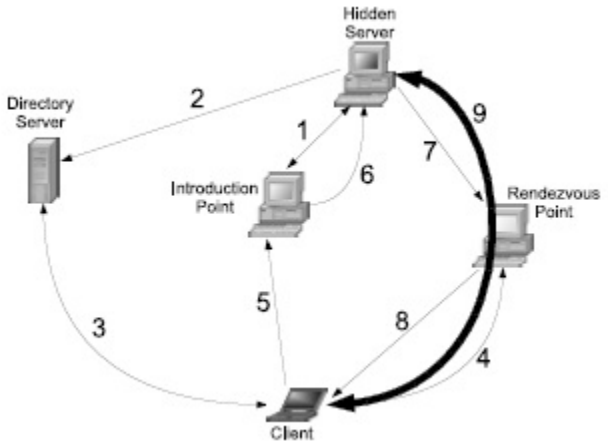

Figure 1: Normal use of hidden services and rendezvous servers.

As described in [5], a normal setup of communication between a client and a hidden service is done as shown in *Figure1*.
First the Hidden Server (HS) connects (1) to a node in the Tor network and asks if it is OK for the node to act as an Introduction Point for its service. If the node accepts, we keep the circuit open and continue. Next, the Hidden Server contacts (2) the Directory Server (DS) and asks it to publish the contact information of its hidden service. In order to retrieve data from the service the client connects (3) to DS and asks for the contact information of the identified service and retrieves it if it exists (including the addresses of Introduction Points).
In our scheme we keep all the circuits of *Figure 1* as such, except for two connections ((2), (3)) concerning the directory server that we modify as follows:
Connection (2): The Hidden Server contacts the Directory Server (DS) and asks it to publish its contact information. The HS will group-sign and timestamp a file $file_1$ where it writes the IP of the introduction points and the name of the Hidden server (as it is done with ζ in section 3.2).
Connection (3): The Client connects to DS asking for the contact information of the identified service (including the addresses of Introduction Points).
In its answer, the DS will send $file_1$ to the client.
Revocation of the anonymity
Let's suppose that the client has a proof showing that this Hidden server is promoting child abuse, the client will exhibit to those who are allowed to revoke the anonymity its proof and $file_1$.
Members of the revocation group (group manager) will revoke this Alice member key's (mk_i) if they are convinced, otherwise they will not.

4 Security analysis

In this scheme, we identify three possible attacks, such as when a user is trying to cheat to remain anonymous in any case. We also have the case of those who try to impersonate some Tor users in order to commit crimes with their identities. Finally, we have attackers who may try to revoke the anonymity of a Tor user, without the consensus of Tor members.

4.1 Attack of the user who wishes to remain anonymous in all cases

The attacker refuses to insert ζ ***or*** $file_1$
In this case, it is in the responsibility of the last node or the DS to reject the Alice's request. It may be noted that if the last relay tries to help Alice to cheat then its liability could be engaged, for example, if it transfers the message to Bob without having received a correct $\zeta \leftarrow Timestamp.GS.sign(< website >: 443, mk_i)$.
Threaten members of the network to prevent the reconstitution of the IP address
In this scheme, due to the properties of group signature, Alice will perhaps be able to identify the different members which can revoke its member key, however these members should be protected against any threat.

4.2 Revocation of the anonymity without the consensus

An attacker may try to revoke the anonymity of Tor users without the consent of Tor's members.
An attacker could guess the circuit by looking into Onion routers (OR)
This attack is impossible, because, even if the attacker sees Alice's group-signature, it cannot guess Alice's member key.
Attack on ζ
In case of an attack, of a legal action or illegal coercion, a node cannot revoke the anonymity alone. Hence, it is impossible to revoke the anonymity without the participation of the group manager.
End-to-end timing correlation
Could the timestamp which we have introduced, help in the End-to-end timing correlation? No, it should not be the case, since the file is not timestamped for the outputting circuit, and between the last relay and the true destination of the packet (Bob) there is no trace of timestamp.

4.3 Impersonate Tor users

The revocation of the anonymity could encourage malicious people, to try to impersonate Tor users by using the following methods.
Spoofing $file_1$ **or** ζ
The attacker cannot use $file_1$ (or ζ) effectively in order to accuse Alice if the connection between Alice and Bob is secured (to avoid a man-in-the-middle attack).
For example, if Alice wants to log on securedemail, she should send
Alice $\rightarrow OR1 : Relay\, c1$ {{ Begin <securedemail.com>:443|ζ}}, where $\zeta \leftarrow Timestamp.GS.sign(< securedemail.com >: 443, mk_i)$.
Then the output node could try to impersonate Alice by connecting to securedemail, but it would not succeed due to the fact that it needs to have Alice's passwords and the fact that the connection between Alice and securedemail is secure against attacks such as the man-in-the-middle.

However, if Alice sends on an unsecured traffic,
Alice $\rightarrow OR1 : Relay\, c1$ {{ Begin <non-secured-web.com>:80|ζ}}, where $\zeta \leftarrow Timestamp.GS.sign(< non-secured-web.com >: 80, mk_i)$, during the time of validity of the timestamp, then the exit node could impersonate Alice in non-secured-web.com, it could also lead to a man-in-the-middle attack, hence the importance of combining Tor with a secure connection.

Other attacks

A malicious node could delay Alice's message in order to create a stop delivery (due to the timestamp).

5 Conclusion

In this paper we showed that the scheme of Diaz et al. was deficient, due to the fact that malicious nodes (essentially the exit nodes) may allow an attacker to impersonate a honest Tor user.
The scheme proposed in order to correct these shortcomings, could also be an improved with additional research on the use of group signatures, which should protect the identity of those who are responsible of the anonymity revocation.

References

[1] Claessens J., Diaz C., Goemans C., Dumortier J., Preneel B., & Vandewalle J. *Revocable anonymous access to the Internet*, Internet Research, 13(4), 242-258, 2003.

[2] Köpsell S., Wendolsky R., & Federrath H. Revocable anonymity. In Emerging Trends in Information and Communication Security, 206-220, 2006.

[3] Kane A. M. *Another Tor is possible* Cryptology ePrint Archive (2014), 787.

[4] Diaz J., Arroyo D., & Rodriguez F. B. *Fair anonymity for the Tor network.* arXiv preprint arXiv:1412.4707, 2014.

[5] Overlier L., Syverson P. *Locating hidden servers.* In Security and Privacy, 2006 IEEE Symposium on (pp. 15-pp).

[6] M. Wright, M. Adler, BN. Levine, C. Shields *An analysis of the degradation of anonymous protocols*, 2002.

[7] Dingledine R., Mathewson N., Syverson P. *Tor: The second-generation onion router.* Naval Research Lab Washington DC, (2004).

[8] Syverson P. *Practical Vulnerabilities of the Tor Anonymity Network.* Advances in Cyber Security: Technology, Operation, and Experiences, (2013).

[9] McCoy D., Bauer K., Grunwald D., Kohno T., Sicker D. *Shining light in dark places: Understanding the Tor network.* In Privacy Enhancing Technologies (2008, January), (pp. 63-76).

[10] Panchenko A., Niessen L., Zinnen A., and Engel T. *Website fingerprinting in onion routing based anonymization networks.* WPES, page 103-114. ACM, 2011.