Fair anonymity for the Tor network

Jesus Diaz
Escuela Politecnica Superior
Universidad Autonoma de Madrid
j.diaz@uam.es

David Arroyo
Escuela Politecnica Superior
Universidad Autonoma de Madrid
david.arroyo@uam.es

Francisco B. Rodriguez
Escuela Politecnica Superior
Universidad Autonoma de Madrid
f.rodriguez@uam.es

ABSTRACT

Current anonymizing networks have become an important tool for guaranteeing users' privacy. However, these platforms can be used to perform illegitimate actions, which sometimes makes service providers see traffic coming from these networks as a probable threat. In order to solve this problem, we propose to add support for *fairness* mechanisms to the Tor network. Specifically, by introducing a slight modification to the key negotiation process with the entry and exit nodes, in the shape of group signatures. By means of these signatures, we set up an access control method to prevent misbehaving users to make use of the Tor network. Additionally, we establish a predefined method for denouncing illegitimate actions, which impedes the application of the proposed fairness mechanisms as a threat eroding users' privacy. As a direct consequence, traffic coming from Tor would be considered less suspicious by service providers.

1. INTRODUCTION

Privacy has become a major concern for Internet users. One main approach for enabling privacy is the introduction of anonymizing techniques, and Tor [17] is probably the most popular and widely used anonymizing network.

Tor anonymizes communications by avoiding origin and recipient to be linked. Moreover, even the recipient cannot learn the IP address of the originator by analyzing the received packets. This is achieved by re-routing the data through several intermediaries, the Onion Routers, and adding an extra layer of encryption with each one. Nevertheless, this also reduces the protection available for the addressee, since it cannot *denounce* the originator or even block him. This is certainly a factor hindering any wide acceptance of anonymizing networks. Moreover, it causes users acting legitimately to be affected by the illegitimate actions of others. For instance, in some situations legitimate users cannot access a site through Tor, because that site directly bans Tor-originated traffic.

In this work we use group signatures to extend the functionality of Tor's entry and exit nodes in order to enable the tracing and blocking of misbehaving users. This being the case, we implement an access control mechanism for Tor which does not deteriorate the normal use of the Tor network by users acting legitimately. As a consequence of this *fairness*[1] mechanism, service providers would probably increase their trust in Tor, since illegitimate actions coming from Tor would presumably be reduced.

Section 2 summarizes some data evidencing that irrevocable anonymity and unlinkability may be seen as a threat, or something undesirable, in some situations, and in Section 3, we summarize some related work. We then describe our approach for incorporating fairness into the Tor network in Section 5. Section 6 points out to some open issues that are important to take into account for realizing our proposal. Finally, Section 7 concludes with a summary of our proposal. Throughout this short paper we assume a basic knowledge of Tor essentials. For a more detailed description of Tor, we refer to [17] or the Tor project website[2].

2. MISTRUST IN COMPLETE ANONYMITY

Despite the fact that an obvious use of communications anonymity is to protect users' privacy, it can also be misused. For instance, in [24] the authors run a Tor exit node for gathering statistical evidence for a further analysis of Tor's traffic. They conclude stating that it is not uncommon to see *"hacking attempts, allegations of copyright infringements, and bot network control channels"* routed through Tor. Another type of behavior that may be considered as undesirable is the use of the Tor platform for purposes other than the originally intended ones, even though they might not be against any ethical rule. This is reported in [10] as a result of analyzing the traffic going through several Tor exit nodes, and concluding that an important share of the packets being routed through Tor corresponds to BitTorrent traffic (roughly the 25%). This reflects that in the case of willing to limit this use of Tor, little more than actually blocking all the BitTorrent traffic could be done. This would be unfair to users routing BitTorrent traffic through Tor but consuming a moderate portion of the available bandwidth. Hence, a finer control on this type of "misbehavior" is also desirable. More evidence supporting the claim that the anonymity provided by Tor can be (and is being) misused is the fact that there already exist services, like BlockScript[3], which include explicit functionality for blocking traffic, including data coming from Tor and other anonymizing networks (BlockScript, offers a commercial service for blocking "unwanted" traffic and sells the raw blacklist data for $12,000 per year). An interesting discussion about the necessity of accountability in anonymous communication systems is done in [14]. Finally, the risk of failure due to websites blocking Tor has also been considered by the Tor staff too in a recent post

[1] According to [22], with *fairness* we refer to the capability of taking any measure that could prevent anonymity misuses.

[2] `https://www.torproject.org`

[3] `http://www.blockscript.com/features.php`.

in the Tor Project blog: *"A call to arms: Helping Internet services accept anonymous users"*[4].

Furthermore there is an abuse specific FAQ in the official Tor website[5] dealing with the subject.

3. RELATED WORK

Several systems have been proposed to endow anonymizing platforms with fairness mechanisms. BLAC [29] makes use of a specific group signature scheme in order to allow service providers to manage their own blacklists, following their own judgment when it comes to block users. Nymble [30] creates a complex infrastructure in order to provide fairness by revoking the unlinkability of users who misbehave. In PEREA [2], the need of Trusted Third Parties is eliminated at the cost of creating a highly crafted infrastructure in order to allow users to be blocked. EPID [7] makes unlinkability revocation possible by either using a member private key, a signature issued by the member to revoke, or by consulting the issuer of the member key. However, it requires the usage of Trusted Privacy Modules [28], which we consider out of the scope of our proposal.

These systems also take into account that different misbehaviors are possible and, in the case of PEREA, that each misbehavior should be assigned a different severity (the PEREA-Naughtiness variant). Nevertheless, they would probably be too rigid for multi-purpose contexts where different legitimate uses and revocation needs are possible. For instance, Nymble just supports unlinkability revocation during a predefined interval. BLAC is tied up to a specific group signature scheme, that might not be appropriate for some situations. PEREA limits the number of authentications that users may perform during a time span, blacklisting them if they exceed that limit. Hence, any change on the implemented functionality that may be necessary for the adoption of these systems into a new context would probably require a great effort.

Our proposal takes advantage of the wide variety of group signatures and the standardization process that anonymous certificates based on them are being subject to [3, 15, 8, 20]. These would allow a high flexibility for modifying the desired functionality depending on the context and, at the same time, the minimization of the deployment costs.

4. BUILDING BLOCKS

We use the notation $\langle O_A, O_B \rangle \leftarrow Pro(I_C)[A(I_A), B(I_B)]$ to describe a two-party process Pro between parties A and B, where O_A (resp. O_B) is the output to party A (resp. B), I_C is the common input, and I_A (resp. I_B) is A's (resp. B's) private input; when party B does not have output, we sometimes write $O_A \leftarrow Pro(I_C)[A(I_A), B(I_B)]$. Single-party processes are denoted by $O \leftarrow Pro(I)$, with input I and output O. Also, for readability, we omit the publicly known keys in the process calls. Finally, since we continuously deal with different types of signatures, we use the Greek letter σ for denote a signature produced by some of the schemes described below (the specific type is clear given the notation in the following subsections).

4.1 Group signatures

Group signatures, first proposed by Chaum and Van Heyst [12], allow a member of a group to issue a signature such that any possible verifier can check that it has been issued by a member of the group, without revealing which specific member issued it. More advanced schemes have been proposed since then [9, 22, 23], improving the scalability and efficiency of group signatures, but also their functionality. Current schemes allow unlinkability revocation and anonymity revocation. Normally, these revocations can only be performed by suitable authorities, but there exist schemes that introduce interesting variants to this behavior, allowing unlinkability revocation only for users who have broken some rule (like exceeding the maximum number of messages to sign [27]).

To summarize, a group of a group signature scheme is basically composed by: a group manager who owns a secret group manager key MK and publishes a group key GK; and a set of N members each in possession of a member key mk_i, for $1 \leq i \leq N$. Overall, the main operations supported by a group signature scheme may be summarized as follows:

$MK, GK \leftarrow \texttt{GS.Setup}(1^k)$**.** Run by the group manager, creates the group key and the manager key.

$mk_i \leftarrow \texttt{GS.Join}[U(secret), M(MK)]$**.** Executed jointly between a new user U and the group manager M, allows new users to join the group, obtaining a member key.

$\sigma \leftarrow \texttt{GS.Sign}(msg, mk_i)$**.** A user creates a group signature over a message, using her member key.

$b \leftarrow \texttt{GS.Verify}(\sigma, msg)$**.** Allows anyone with the group key to verify a group signature.

$trapdoor \leftarrow \texttt{GS.Open}(\sigma, MK)$**.** Run by the group manager, allows him to obtain de-anonymize to some extent the issuer of a group signature, given the group signature itself, the manager key and, normally, some additional private information. In some schemes, the only possibility is to retrieve her real identity. In others, it is possible to obtain a token which permits to link group signatures made by the same user. For simplicity, in this work we encompass all variants that somehow reduce anonymity under the term `Open`.

$b \leftarrow \texttt{GS.Trace}(\sigma, MK)$**.** Allows to verify if a given group signature has been issued by some arbitrary member. In order to run this operation, `GS.Open` needs to be executed before. Thus, note that there may be different ways to achieve this, according to the variants of the `GS.Open` functionality.

However, this can be considered a "general working mode" of a group signature scheme. For a good review of the main advancements in the field of group signatures, we refer to [23, Section 1.1].

4.2 Blind signatures

Blind signatures where introduced by Chaum in [11]. Basically, a blind signature scheme allows a user U to obtain a signature from a signer S over any arbitrary message m, but without S learning anything about m. This idea has been used since then for creating diverse privacy respectful systems. However, since the signer does not learn any information about the message, systems based on them can

[4] https://blog.torproject.org/blog/call-arms-helping-internet-services-accept-anonymous-users.

[5] https://www.torproject.org/docs/faq-abuse.html.en.

easily be abused. For solving this issue, *fair blind signatures* were proposed in [26]. In this variant, an authority has privileged information allowing the signer to link message and signature pairs. *Restrictive blind signatures* [6] allow issuing blind signatures, but only choosing among messages that comply certain rules. Finally, an also important alternative is given by partially blind signatures [1]. In a *partially blind signature* the messages are divided in two parts: a common message to which the S has complete access; and the blinded message, of which S does not learn anything. Thus, the common message may be employed to implement misuse prevention mechanisms. As always, several schemes have appeared improving the overall efficiency, reducing the size of the final signatures, or based on different number theory problems [21, 13, 25, 5].

Although some of the variants of blind signatures would probably be useful for our proposal, we use the general definition of a blind signature for describing it. Thus, hereafter we will use the following notation when referring to the operations of a blind signature scheme:

$(pbk, sbk) \leftarrow \texttt{BS.Setup}(1^k)$**.** Creates the signer's public pbk and private keys sbk for issuing blind signatures.

$(\beta, \pi) \leftarrow \texttt{BS.Blind}(msg, secret)$**.** Using some random secret value, the user creates a blinded version (β) of the message to be blindly signed and a proof of correctness π.

$\tilde{\beta} \leftarrow \texttt{BS.Sign}(\beta, sbk)$**.** Upon receiving the blinded messages, the signer runs any necessary verification and creates a blinded signature using its private key.

$\sigma \leftarrow \texttt{BS.Unblind}(\tilde{\beta}, secret)$**.** The user receives the blinded signature and unblinds it, using the secret value generated during the blind process. The result of this operation is the final signature.

$b \leftarrow \texttt{BS.Verify}(\sigma, msg)$**.** Any entity runs this operation to verify the signature.

4.3 Blind group signatures

A blind group signature scheme is just like a blind signature in which the signer issues a group signature instead of a conventional signature. Therefore, each of the operations described in the previous sections may be independently applied in this schemes. However, for the sake of clarity, when referring to this schemes, we will use the prefix `BGS` instead of `GS` or `BS`.

4.4 Additional cryptographic primitives

Besides the primitives introduced above, we assume readers are familiar with public-key encryption, digital signature and commitment schemes, and zero-knowledge proofs of knowledge. We use $\texttt{com} \leftarrow \texttt{Com}(m, r)$ to denote a commitment `com` to a message m, where the sender uses uniform random coins r; the sender can open the commitment by sending (m, r) to the receiver. We use $\pi \leftarrow \texttt{ProveZK}(x, w)$ and $\texttt{VerifyZK}(x, \pi)$ to refer to creating non-interactive proof π showing that the statement x is in the language (which will be determined by the context) with the witness w, and to verifying the statement x based on the proof π.

5. GUIDELINES FOR INCORPORATING FAIRNESS INTO TOR

In order to endow Tor with fairness capabilities, the entry and exit nodes take a central role, since they are the only nodes who learn the IP addresses of the user entering the network and that of the final destination, respectively. Hence, their knowledge would be necessary to determine whether the IP trying to access the network has already been blocked, or to demonstrate that a given origin IP has accessed certain destination IP. However, when proposing modifications of those nodes we must avoid enabling attacks based on establishing a connection between them. For that purpose, we take advantage of both the way the user negotiates keys with the Tor nodes, and the properties of group and blind signatures.

Hereafter, we assume that a group has already been set up, and that there is a suitable policy established for fairly managing revocation (see Section 6). Similarly, we assume that the blind signature scheme has also been set up. Table 1 summarizes the notation used throughout the rest paper, along with some notation inherited from the description of the Tor network [17] and the notation defined for group and blind signatures and the additional cryptographic primitives in Section 4.

$\{\cdot\}_K$	Symmetric encryption with key K
$\{\cdot\}_{PK_A}$	Asymmetric encryption with A's public key.
$\{\{\cdot\}\}$	Layered encryptions following the Tor protocol.
$H(\cdot)$	Application of a cryptographic hash function.
$g^{x\cdot}$	The user's Diffie-Hellman share.
$g^{y\cdot}$	The Diffie-Hellman share corresponding to a Tor node.
hs_K	A transcription of the handshake for key K.
$A \mid B$	A concatenated with B.
σ_1	Group signature of g^{x_1} sent to entry node.
σ_2	Group signature of g^{x_2} sent to exit node.
β	Blinded version of σ_2
$\tilde{\beta}$	Blindly signed version of β
σ_3	Blind signature of σ_2

Table 1: Notation summary.

Our approach works by introducing variations in the way a user negotiates the symmetric keys with the entry and exit nodes. In short, we will require the user to group-sign the message sent during negotiation with the entry and exit nodes. 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 (see Section 6), the entry node has to blindly sign the message that the user will send to the exit node. The resulting modified handshake schemes (see [17, p. 6]) are shown below, where $\texttt{U}_i$ denotes any arbitrary user, `EN` denotes the entry node and `EX` the exit node. During the handshake with `EN`, $\texttt{U}_i$ 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.

Entry Node Handshake:

```
$\mathtt{U}_i$: $\sigma_1 \leftarrow \mathtt{GS.Sign}(g^{x_1}, mk_i)$
$\mathtt{U}_i$: $\sigma_2 \leftarrow \mathtt{GS.Sign}(g^{x_2}, mk_i)$
$\mathtt{U}_i$: $\mathsf{com} \leftarrow \mathsf{Com}(\sigma_2, r_1)$
$\mathtt{U}_i$: $(\beta, \pi) \leftarrow \mathtt{BGS.Blind}(\mathtt{com}, r_2)$
$\mathtt{U}_i$: $\phi \leftarrow \mathtt{ProveZK}(x, w)$ where
  $x = (\beta, \pi, \sigma_1), w = (mk_i, r_1, r_2)$ such that:
    $\sigma_2 \leftarrow \mathtt{GS.Sign}(g^{x_2}, mk_i)$,
    $(\beta, \pi) \leftarrow \mathtt{BGS.Blind}(\mathsf{Com}(\sigma_2, r_1), r_2)$
$\mathtt{U}_i \rightarrow \mathtt{EN}$: $g^{x_1}, \sigma_1, \beta, \pi, \phi$
$\mathtt{EN}$: $\mathtt{VerifyZK}(\beta, \pi, \phi, \sigma_1)$
$\mathtt{EN}$: $\mathtt{GS.Verify}(\sigma_1, g^{x_1})$
$\mathtt{EN}$: $\tilde{\beta} \leftarrow \mathtt{BGS.Sign}(\beta, sbk)$
$\mathtt{EN}$: $K_1 = g^{x_1 y_1}$
$\mathtt{EN} \leftarrow \mathtt{U}_i$: $g^{y_1}, \tilde{\beta}, H(K_1|hs_{K_1})$
$\mathtt{U}_i$: $\sigma_3 \leftarrow \mathtt{BGS.Unblind}(\tilde{\beta}, r_2)$
$\mathtt{U}_i$: $K_1 = g^{x_1 y_1}$
```

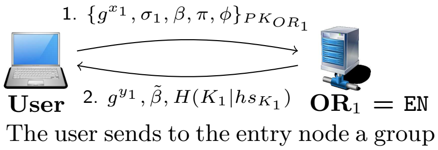
When $\mathtt{U}_i$ initiates the handshake with $\mathtt{EX}$, she sends the group signature on g^{x_2} that was blindly signed by $\mathtt{EN}$, along with the blind signature itself. If all the verifications succeed, then $\mathtt{EX}$ accepts the connection.

Exit Node Handshake:

```
$\mathtt{U}_i \rightarrow \mathtt{EX}$: $g^{x_2}, \sigma_2, \sigma_3$
$\mathtt{EX}$: $\mathtt{GS.Verify}(\sigma_2, g^{x_2})$
$\mathtt{EX}$: $\mathtt{BGS.Verify}(\sigma_3, \sigma_2)$
$\mathtt{EX}$: $K_2 = g^{x_2 y_2}$
$\mathtt{EX} \rightarrow \mathtt{U}_i$: $g^{y_2}, H(K_2|hs_{K_2})$
$\mathtt{U}_i$: $K_2 = g^{x_2 y_2}$
```

It is important to note that the group signatures are encrypted using the public keys of either the entry or exit nodes. Hence, only the entry and exit nodes learn them. Moreover, the group signature sent to the exit node is blindly signed by the entry node. Thus, even if both nodes collude, they would not be able to determine by themselves that the group signatures they have received have been issued by the same user, due to the unlinkability property of the group signature scheme and the blindness property of the blind signature scheme. Moreover, since the group signature sent to the exit node has been blindly signed by the entry node, it is not possible for a user $\mathtt{U}_i$ to frame another user $\mathtt{U}_j$.

The modified key negotiation with the entry node is depicted in Fig. 1, and the one corresponding to the exit node is depicted in Fig. 2.

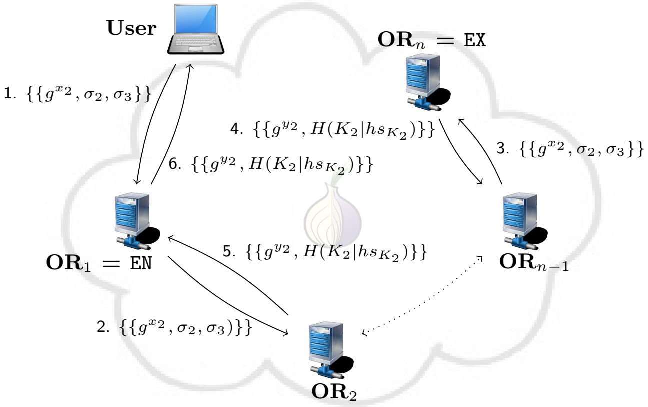


1. $\{g^{x_1}, \sigma_1, \beta, \pi, \phi\}_{PK_{OR_1}}$
User
2. $g^{y_1}, \tilde{\beta}, H(K_1|hs_{K_1})$
$\mathbf{OR}_1 = \mathtt{EN}$

Figure 1: The user sends to the entry node a group signature of her share of the key, encrypted with the node's public key, and a blinded version of the group signature to be sent to the exit node. The entry node returns a blindly signed version of the latter.

5.1 How to block misbehaving users

Let us assume that some user $\mathtt{U}_i$ has been revoked due to some illegitimate behavior. When $\mathtt{U}_i$ tries to establish a circuit, he/she will need to perform a handshake with the chosen Tor entry node. Hence, upon receiving the first message with the group signature, the entry node will verify the received group signature, checking whether or not the member who issued it has been revoked. Given that the member key of $\mathtt{U}_i$ has been revoked, the verification will fail, and the entry node will reject the connection. Note that if the user has not been revoked, the privacy guarantees provided by Tor are not diminished.



User
$\mathbf{OR}_n = \mathtt{EX}$
1. $\{\{g^{x_2}, \sigma_2, \sigma_3\}\}$
4. $\{\{g^{y_2}, H(K_2|hs_{K_2})\}\}$
3. $\{\{g^{x_2}, \sigma_2, \sigma_3\}\}$
6. $\{\{g^{y_2}, H(K_2|hs_{K_2})\}\}$
$\mathbf{OR}_1 = \mathtt{EN}$
5. $\{\{g^{y_2}, H(K_2|hs_{K_2})\}\}$
$\mathbf{OR}_{n-1}$
2. $\{\{g^{x_2}, \sigma_2, \sigma_3)\}\}$
$\mathbf{OR}_2$

Figure 2: The user sends to the exit node a group signature of her share of the key, encrypted with the node's public key and the blind signature issued by the entry node.

5.2 How to denounce misbehaving users

In this case, we assume that $\mathtt{U}_i$ has already established a circuit and she is communicating with some server S (external to Tor). Also, let us suppose that eventually, $\mathtt{U}_i$ performs some illegitimate action. When that happens, S denounces this behavior following some predefined method. If deemed appropriate, the group signature received by the exit node during the handshake may be used to retrieve $\mathtt{U}_i$'s identity, or to trace her. Specifically, the exit node provides the following information:

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

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 ($\mathtt{U}_i$) is responsible of msg. Hence, $\mathtt{U}_i$'s key can be consequently revoked, and the circuit may be closed by the exit node. Note that subsequent attempts made by $\mathtt{U}_i$ to establish a circuit would be blocked by the entry node, since the member key of $\mathtt{U}_i$ has been revoked.

5.3 Additional remarks

A few remarks are worth to be made, concerning "special" situations and subjects that should be taken into account.

Tor bridges Our proposal is directly extensible to support Tor bridges, considering Tor bridges as the entry nodes to the Tor network.

Leaky pipe topology This approach allows a client to output some packets through a node other than the negotiated exit. In order to adapt our approach to this exception, the node leaking the packet should follow ad-hoc the procedure defined for exit nodes.

Logging Since hosts being accessed through these *fair* Tor extension might want to issue denounces against traffic originated from Tor, logging the information necessary for solving disputes is required. Hence, the exit nodes need to keep the information specified in Section 5.2 (this also applies to the leaky pipe extension). An appropriate policy for logging should be established.

Denunciation time span Considering that our proposal requires Tor exit nodes (and circumstantially other nodes) to log several pieces of information, it would comprise a serious scalability problem if this logging would be expected to last too much time. Hence, it seems appropriate to establish a predetermined time span for accepting denounces. Upon expiration of that time span, all the logged information could be removed, and any subsequent denounce related to that information rejected. This would require possible complainants to be aware of this time limitation.

6. OPEN ISSUES

In the scheme given in Section 5 we just use the general definitions of the building blocks for defining our system. The analysis of which specific variants should be employed is left as future work. Note that this is a very delicate decision, since different options offer different privacy properties. Moreover, we may even need different schemes depending on who issues the signatures (e.g. group signatures are issued both by users and entry points in our proposal). Thus, given the sensitivity of the information managed by Tor, this is an issue that needs to be studied in depth by itself. For that matter, the extensible group signatures library `libgroupsig` [16] may offer interesting features. In addition, concerning the blind signatures, it would probably be necessary to use some of its variant to prevent circumventing the controls explaining above. Namely, with the previous bare scheme, a user could use the same blind signature indefinitely. This may simply be solved by using partially blind signatures, and having the entry node introduce a *lifetime* value for the blind signature as common message.

Another important issues are determining when misbehaving users should be revoked, and by whom. The former question would probably depend on the websites (or service) being accessed through Tor. For the latter, a probably good solution given Tor's infrastructure would be to apply threshold schemes to the revocation procedures (see [4]), such that a majority of the authorities participating in the network consensus need to agree for revoking users.

Finally, note that Sybil attacks [18] are partly addressed by forcing users to use the same member key for the group signature sent to the entry node and for the group signature sent to the exit node (and having the latter to be blindly signed by the entry node). However, some additional mechanism should be included for preventing users from arbitrarily generating new member keys. Since asking users to register may not be well received (it may be seem as a threat to anonymity), requesting them to perform some proof of work [19] during the generation of the member keys may be a good alternative.

7. CONCLUSION

In this work we have proposed an extension to the Tor network in order to endow it with the functionality for preventing misbehaving users to access the network. We expect such functionality to increase the trust of websites in Tor and thus prevent them to block users coming from it. This extension follows the design of Tor, and does not require any modification to its infrastructure. It works by including two group signatures in the key negotiation processes with the entry and exit nodes (one for each) and having the entry node blindly sign the group signature to be sent to the exit node. The group signature sent to the exit node allows service providers to denounce illegitimate actions. Once the unlinkability of a user has been revoked as a consequence of some illegitimate behavior, any entry node would be able to block that specific user just by checking if it is included in an (unlinkability) Revocation List.

References

[1] M. Abe and E. Fujisaki. How to date blind signatures. In *ASIACRYPT*, pages 244–251, 1996.

[2] M. H. Au, P. P. Tsang, and A. Kapadia. PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. *ACM Trans. Inf. Syst. Secur.*, 14(4):29, 2011.

[3] V. Benjumea, S. G. Choi, J. Lopez, and M. Yung. Anonymity 2.0 - X.509 extensions supporting privacy-friendly authentication. In *CANS*, pages 265–281, 2007.

[4] V. Benjumea, S. G. Choi, J. Lopez, and M. Yung. Fair traceable multi-group signatures. In *Financial Cryptography*, pages 231–246, 2008.

[5] O. Blazy, G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Short blind signatures. *Journal of Computer Security*, 21(5):627–661, 2013.

[6] S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *CRYPTO*, pages 302–318, 1993.

[7] E. Brickell and J. Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *Dependable and Secure Computing, IEEE Transactions on*, 9(3):345–360, 2012.

[8] J. Camenisch, I. Krontiris, A. Lehmann, G. Neven, C. Paquin, K. Rannenberg, and H. Zwingelberg. D2.1 architecture for attribute-based credential technologies - version 1, 2011.

[9] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76, 2002.

[10] A. Chaabane, P. Manils, and M. A. Kâafar. Digging into anonymous traffic: A deep analysis of the tor anonymizing network. In *NSS*, pages 167–174, 2010.

[11] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[12] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[13] X. Chen, F. Zhang, Y. Mu, and W. Susilo. Efficient provably secure restrictive partially blind signatures from bilinear pairings. In *Financial Cryptography*, pages 251–265, 2006.

[14] C. Diaz and B. Preneel. Accountable anonymous communication. In *Security, Privacy, and Trust in Modern Data Management*, Data-Centric Systems and Applications, pages 239–253. Springer Berlin Heidelberg, 2007.

[15] J. Diaz, D. Arroyo, and F. B. Rodriguez. New x.509-based mechanisms for fair anonymity management. *Computers & Security*, 46:111–125, 2014.

[16] J. Diaz, D. Arroyo, and F. B. Rodriguez. `libgroupsig`: an extensible c library for group signatures. *submitted*, 2015.

[17] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320, 2004.

[18] J. R. Douceur. The sybil attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, pages 251–260, 2002.

[19] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 139–147, 1992.

[20] ISO/IEC. ISO/IEC CD 20008-1.2: Information technology - security techniques - anonymous digital signatures - part 1: General, 2012.

[21] W.-S. Juang and C.-L. Lei. Partially blind threshold signatures based on discrete logarithm. *Computer Communications*, 22(1):73–86, 1999.

[22] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT*, pages 571–589, 2004.

[23] B. Libert, T. Peters, and M. Yung. Group signatures with almost-for-free revocation. In *CRYPTO*, pages 571–589, 2012.

[24] D. McCoy, K. S. Bauer, D. Grunwald, T. Kohno, and D. C. Sicker. Shining light in dark places: Understanding the Tor network. In *Privacy Enhancing Technologies*, pages 63–76, 2008.

[25] T. Okamoto. Efficient blind and partially blind signatures without random oracles. In *TCC*, pages 80–99, 2006.

[26] M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *EUROCRYPT*, pages 209–219, 1995.

[27] I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication (extended abstract). In *ASIACRYPT*, pages 308–322, 2004.

[28] Trusted Computing Group. TCG TPM Specification 1.2. `http://www.trustedcomputinggroup.org`, 2003.

[29] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *ACM Conference on Computer and Communications Security*, pages 72–81, 2007.

[30] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith. Nymble: Blocking misbehaving users in anonymizing networks. *IEEE Trans. Dependable Sec. Comput.*, 8(2):256–269, 2011.