Fair and Accountable Anonymity for the Tor Network

Jesus Diaz[1], David Arroyo[2] and Francisco B. Rodriguez[2]
[1]*BEEVA, Madrid, Spain*
[2]*Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain*

Keywords: Privacy Enhancing Technologies, Group Signatures, Blind Signatures.

Abstract: The balance between security and privacy is a must for the adequate construction of e-democracy. For such a goal, information and communication networks should not be hardened at the cost of lessening privacy protection mechanisms. In addition, the deployment of such mechanisms should not pave the way for performing malicious activities. This call for security-privacy trade-offs is specially relevant in the general scope of the anonymizing networks, and in the specific case of the Tor network. Indeed, general security attacks are based on anonymous network access, which makes service providers to ban this kind of connections even when they are initiated by legitimate users. In this communication we apply group and blind signatures to address this dilemma, allowing the incorporation of access controls to the Tor network. Our procedure is enhanced by a protocol for denouncing illegitimate actions without eroding users' privacy.

1 INTRODUCTION

One main approach for enabling privacy is the introduction of anonymizing techniques, and Tor Dingledine et al. (2004) is probably the most popular and widely used option. It anonymizes communications by re-routing data packets through several intermediaries, the Onion Routers, adding an extra layer of encryption with each one. Nevertheless, this makes the recipient vulnerable to attacks, since it cannot *denounce* the originator or even block him. This is a factor hindering a wide acceptance of anonymizing networks. Moreover, it causes legitimate users to be affected by the illegitimate actions of others. For instance, sometimes legitimate users cannot access a site through Tor, because that site directly bans Tor-originated traffic. An interesting discussion about the necessity of accountability in anonymous communication systems is done in Diaz and Preneel (2007). The risk derived from websites blocking Tor[1] has also been considered by the Tor staff[2].

In this work we use group signatures to extend the functionality of Tor's entry and exit nodes in order to enable tracing and blocking dishonest users. We actually 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*[3] mechanism, service providers would probably increase their trust in Tor, since illegitimate actions coming from Tor would presumably be reduced.

2 RELATED WORK

Several systems have been proposed to endow anonymizing platforms with fairness mechanisms. BLAC Tsang et al. (2007) makes use of a specific group signature scheme for the sake of users blacklisting on the grounds of self-established criteria. Nymble Tsang et al. (2011) achieves fairness by revoking the unlinkability of misbehaving users. In PEREA Au et al. (2011), users blocking is fulfilled without Trusted Third Parties at the cost of creating a highly crafted infrastructure that must be built on top of an anonymizing network. PE(AR)2 Yu et al. (2012) improves PEREA's efficiency and offers a more advanced and builtin reputation system. EPID Brickell and Li (2012) supports several types of anonymity revocation. However, it requires the usage of trusted

[1]See, for instance: https://blog.torproject.org/blog/trouble-cloudflare.

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

[3]As in Kiayias et al. (2004), with *fairness* we refer to the capability of taking measures for preventing anonymity misuses.

Diaz, J., Arroyo, D. and Rodriguez, F.
Fair and Accountable Anonymity for the Tor Network.
DOI: 10.5220/0006474805600565
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 560-565
ISBN: 978-989-758-259-2

hardware modules, which we consider a too strict requirement.

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. 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.

Besides the previous observations, the main difference between our system and the previous ones is a design choice. These systems introduce accountability at the application layer. Thus, dishonest users still have access to the underlying anonymizing network and can still collapse it, with the consequent reduction of anonymity protection to legitimate users. Our design works at the network layer, directly preventing misbehaving users to access the Tor network. Furthermore, note that it is possible to convey accountability information from the network layer to the application layer, hence providing the same functionality to the final server. Finally, our proposal leverages the wide variety of group signatures and the standardization process of anonymous certificates based on them Diaz et al. (2014). These allow high flexibility for adapting the functionality depending on the context as well as the minimization of the deployment costs.

3 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.

3.1 Group Signatures

Group signatures, proposed by Chaum and Van Heyst Chaum and van Heyst (1991), 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. Advanced schemes have been proposed since, improving the scalability and efficiency of group signatures, but also their functionality Kiayias et al. (2004); Libert et al. (2012). 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:

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

$mk_i \leftarrow$ **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.

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

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

$trapdoor \leftarrow$ **GS.Open**(σ, 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 our system, we assume group signature schemes that also allow privacy respectful traceability, called traceable signatures Kiayias et al. (2004). These systems include an additional operation:

$b \leftarrow$ **GS.Trace**(σ, 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.

3.2 Blind Signatures

Blind signatures where introduced by Chaum Chaum (1982). 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*. As usual, variants of the original blind signature proposal have appeared, offering additional functionality or improving its efficiency Brands (1993); Abe and Fujisaki (1996); Blazy et al. (2013). Although some

of these variants would probably be useful for our proposal, we use the general definition of a blind signature for describing it, according to the following notation:

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

$(\beta, \pi) \leftarrow$ **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.

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

$\sigma \leftarrow$ **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 is the final signature.

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

We also use combination of group and blind signatures: blind group signatures. A blind group signature is like a blind signature where the signer issues a group signature instead of a conventional signature. For the sake of clarity, when referring to this schemes, we use the prefix BGS.

3.3 Additional Primitives

Besides the primitives introduced above, we assume readers are familiar with public key encryption, digital signature and commitment schemes, and zero-knowledge proofs. We use $\mathtt{com} \leftarrow \mathsf{Com}(m, r)$ 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 $\pi \leftarrow \mathtt{ProveZK}(x, w)$ and $\mathtt{VerifyZK}(x, \pi)$ 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 witness w, and to verifying statement x based on proof π.

4 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 directly connected to origin and recipient of the communication, respectively. However, when proposing modifications 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 group and blind signature schemes have already been set up, and that there is a policy established for fairly managing revocation (see Section 5). Table 1 summarizes the notation used throughout the rest paper, along with some notation inherited from the description of the Tor network Dingledine et al. (2004) and the notation defined for group and blind signatures and the additional cryptographic primitives in Section 3.

Table 1: Notation summary.

$\{\cdot\}_K$	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 (DH) share.
$g^{y_\cdot}$	The DH share corresponding to a Tor node.
hs_K	A transcription of the handshake for key K.
$A\|B$	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

Our approach works by introducing variations in the key negotiation between users with entry and exit nodes. We require users to group-sign the message sent during negotiation with entry and exit nodes. In addition, for preventing users to employ one identity for negotiating with the entry node, and a different one with the exit node, zero knowledge proofs tying up both signatures are included. The modified handshake schemes (see (Dingledine et al., 2004, p. 6)) 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 group-signs g^{x_1} and g^{x_2}, sends g^{x_1} to EN and requests EN to blindly sign a group signature of g^{x_2}. If all operations succeed, EN accepts the connection. The blind signature is needed because it will be shown to the EX. Thus, a blind signature prevents colluding EN and EX to detect they are both in the same circuit.

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

When U_i initiates the handshake with EX, she sends the group signature on g^{x_2} that was blindly signed by EN, and the blind signature itself. If all checks succeed, EX accepts the connection.

Exit Node Handshake:
$\mathsf{U}_i \rightarrow \mathsf{EX}$: $g^{x_2}, \sigma_2, \sigma_3$
EX: $\mathtt{GS.Verify}(\sigma_2, g^{x_2})$
EX: $\mathtt{BGS.Verify}(\sigma_3, \sigma_2)$
EX: $K_2 = g^{x_2 y_2}$
$\mathsf{EX} \rightarrow \mathsf{U}_i$: $g^{y_2}, H(K_2 | hs_{K_2})$
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 U_i to frame another user 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.

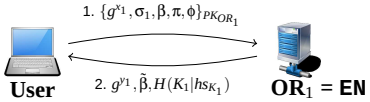

Figure 1: The user sends to the entry node a group signature of her share of the key, encrypted with the node's public key, and a blinded version of the group signature to be sent to the exit node. The entry node returns a blindly signed version of the latter.

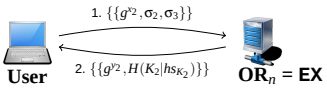

Figure 2: The user sends to the exit node a group signature of her share of the key and the blind signature by the entry node, encrypted with the exit node's public key. Note that these messages are routed through a Tor circuit composed of several nodes.

4.1 Blocking Misbehaving Users

Let us assume that some user U_i has been revoked due to some illegitimate behavior. When 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 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. Since, as pointed out in Section 3.1, we use traceable (group) signatures, checking for revoked users can be done in a privacy respectful manner (i.e., without opening group signatures).

4.2 Denouncing Misbehaving Users

In this case, we assume that U_i has already established a circuit and she is communicating with some server S (external to Tor). We also assume that there is a Revocation Authority (RA) that decides whether or not an action is illegitimate and can open group signatures in that case.

Now, suppose that eventually U_i performs some illegitimate action. Then, S denounces this behavior following some predefined method. If deemed appropriate, the group signature received by EX during the handshake may be used to retrieve U_i's identity, or to trace her. Specifically, EX provides RA the following information:

- $\{msg\}_K$, where msg is the message denounced by S, and K is the symmetric key negotiated between U_i and EX.
- $(K = g^{x_2 y_2}, g^{x_2})$, where g^{x_2} is U_i's share of the handshake and g^{y_2} is the share created by EX.
- σ_2, i.e., a group signature of g^{x_2} issued by U_i.
- σ_3, i.e., the blind group signature over σ_2 issued by EN.

In order to verify that the received denounce is valid, it is necessary to check that the message re-

ceived from *S*, *msg*, corresponds to the encryption $\{msg\}_K$ received from the exit node. Also, σ_3 must be a valid group signature over σ_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 either:

- σ_2 is not a valid group signature over g^{x_2}, or it is a valid group signature, but issued by a revoked user: then EN misbehaved. In this case, RA opens σ_3 and proceeds according to some defined policy for misbehaving entry nodes.
- σ_2 is a valid group signature over g^{x_2}, issued by an unrevoked user: then RA opens it and revokes the associated user.

Subsequent attempts by U_i to establish a circuit would be blocked by EN, since the member key of U_i has been revoked. Also, the pieces of information needed to denounce a user could be stored temporarily by EX, or sent to *S*. In the former case, if *S* wants to denounce a misbehavior and EX does not have a copy of the required data, then EX misbehaved (policies for how much time EX is required to store the data should be defined). In the latter case, EX should just send this data to *S*. Even if malicious EN or EX nodes share with malicious users the group signatures obtained from honest users, the malicious users cannot re-generate valid zero-knowledge proofs. This will lead to the detection of dishonest EN and EX nodes, and their revocation.

5 OPEN ISSUES

In Section 4 we use generic definitions of the building blocks. The analysis of which specific variants should be employed is left as future work. 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 (group signatures are issued both by users and entry points in our proposal). Given the sensitivity of the information managed by Tor, this is an issue that needs to be studied in depth. For that matter, the extensible group signatures library `libgroupsig` Diaz et al. (2015) may offer interesting features. Concerning blind signatures, it would probably be necessary to use some of its variants 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, having the entry node introduce a *lifetime* value for the blind signature as common message. Of course, future work comprises the formal definition of the security model of our proposal and the verification of its security claims.

Finally, note that Sybil attacks 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, since this setting requires dynamic groups, 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 Dwork and Naor (1992) during the generation of the member keys may be a good alternative. Following this line of trying to reduce the trust users would need to place in a system that, by introducing accountability, can be at most as anonymous as a fully anonymous sytem, we can include secret sharing techniques to divide the capability of user revocation among several authorities Benjumea et al. (2008). Furthermore, it would be possible to incorporate contractual anonymity (or objective blacklisting) techniques, to prevent certain unjustified revocations Schwartz et al. (2010).

5.1 Efficiency

Concerning the efficiency of the proposed scheme, it is reasonable to ask whether or not the additional cryptographic operations would incur in an acceptable cost. This is obviously a relevant future line of work. However, note that the additional operations need to be executed only once each time a circuit is established[4], which only occurs every 10 minutes. The proposed extension requires the user to compute two group signatures, a ZK proof and interact with the entry node for issuing a blind group signature. The entry and exit nodes verify one and two group signatures, respectively. According to the experiments done in Diaz et al. (2015), sign and verify operations of traceable group signatures take approximately 0.05 seconds with ECC-based keys of up 256 bits. Thus, the overload associated to group signatures should not take more than 0.3 seconds per circuit. For the ZK proof and blind part of the blind group signature we have not found experimental measurements, but it is likely to be in the same order. Thus, the additional cost seems quite bearable.

[4]Excluding the operations for setting up the group of users and generation of group member keys, but these are expected to occur very infrequently.

6 CONCLUSION

In this work we proposed an extension to the Tor network for preventing dishonest users to access the network. This effort is intended to foster websites' trust on Tor and avoid Tor traffic filtering as a general protection procedure. Our extension follows the design of Tor, and does not require any modification to its infrastructure. It works by slightly modifying the handshake performed with entry and exit nodes, including group and blind signatures. Our proposal allows entry nodes to block dishonest users. Thus, revoked users do not even consume Tor's bandwidth beyond a (failed) handsake with an entry node.

ACKNOWLEDGEMENTS

The work of David Arroyo was supported by Comunidad de Madrid (Spain) under the project S2013/ICE-3095-CM (CIBERDINE). The work of Francisco B. Rodriguez was supported by the Spanish Government project TIN2014-54580R.

REFERENCES

Abe, M. and Fujisaki, E. (1996). How to date blind signatures. In *ASIACRYPT*.

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

Benjumea, V., Choi, S. G., Lopez, J., and Yung, M. (2008). Fair traceable multi-group signatures. In *Financial Cryptography*.

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

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

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

Chaum, D. (1982). Blind signatures for untraceable payments. In *CRYPTO*.

Chaum, D. and van Heyst, E. (1991). Group signatures. In *EUROCRYPT*, pages 257–265.

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

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

Diaz, J., Arroyo, D., and Rodriguez, F. B. (2015). `libgroupsig`: an extensible c library for group signatures.

Dingledine, R., Mathewson, N., and Syverson, P. F. (2004). Tor: The second-generation onion router. In *USENIX Security Symposium*.

Dwork, C. and Naor, M. (1992). Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO '92*.

Kiayias, A., Tsiounis, Y., and Yung, M. (2004). Traceable signatures. In *EUROCRYPT*, pages 571–589.

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

Schwartz, E. J., Brumley, D., and McCune, J. M. (2010). Contractual anonymity. In *NDSS*.

Tsang, P. P., Au, M. H., Kapadia, A., and Smith, S. W. (2007). Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *ACM CCS*, pages 72–81.

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

Yu, K. Y., Yuen, T. H., Chow, S. S. M., Yiu, S., and Hui, L. C. K. (2012). PE(AR)2: privacy-enhanced anonymous authentication with reputation and revocation. In *Computer Security - ESORICS 2012, Italy. Proceedings*.