FAUST: Efficient, TTP-Free Abuse Prevention by Anonymous Whitelisting

Peter Lofgren
Stanford University
peter@lofgren.us

Nicholas Hopper
University of Minnesota
hopper@cs.umn.edu

ABSTRACT

We introduce Faust, a solution to the "anonymous blacklisting problem:" allow an anonymous user to prove that she is authorized to access an online service such that if the user misbehaves, she retains her anonymity but will be unable to authenticate in future sessions. Faust uses no trusted third parties and is one to two orders of magnitude more efficient than previous schemes without trusted third parties. The key idea behind Faust is to eliminate the explicit blacklist used in all previous approaches, and rely instead on an implicit whitelist, based on blinded authentication tokens.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Authentication*; E.3 [**Data Encryption**]: *Public key cryptosystems*

General Terms

Algorithms, Security

Keywords

Anonymous Authentication, Anonymous Blacklisting, Privacy-Enhancing Revocation

1. INTRODUCTION

Anonymity networks like Tor [7], UltraSurf [6] and JonDo [8] allow users to access online services while concealing the parties to any particular communication, by relaying this information through one or more intermediaries. These networks help to circumvent online censorship and protect freedom of speech, but represent a "mixed blessing" for the providers of online services. In particular, anonymous access can expand the range of users that contribute to an online service, but it can also allow misbehaving users to abuse the service in a way that makes it difficult to hold them accountable. As a result, several service providers – including Wikipedia, Slashdot, CraigsList, and several IRC channels – have chosen to block contributions from known anonymity providers, despite the implied loss of contributions.

To address this problem, Johnson *et al.* [14] (inspired by [13]) proposed the notion of an *anonymous blacklisting scheme*, which allows service providers (SPs) to maintain a "blacklist" such that non-abusive users can access the service anonymously; while users on the blacklist cannot access the service, but remain anonymous. Several such schemes have appeared in the literature, dealing with blacklist management in one of two ways. In "Nymble schemes" [14, 24, 11, 15, 16, 9], a set of trusted third parties (TTPs) manage the blacklist, allowing efficient authentication and blacklisting but making both the SP and the user dependent on the TTPs for privacy and blacklistability. In contrast, TTP-free schemes [21, 4, 3, 22] provide privacy and blacklistability by construction but require the user and SP to perform expensive zero-knowledge proofs to show that a user is not blacklisted; the computational complexity (per user) of these schemes scales poorly with the number of users.

Our Contribution. In this paper, we introduce Faust, a TTP-Free scheme that eliminates the blacklist and replaces it with an implicit "whitelist:" each user receives a "token" that proves her last action was acceptable; when she connects to the SP again, she reveals this token and sends a blinded token that she can privately retrieve after a period of time deemed sufficient to detect abuse. If her action is deemed unacceptable, no special action is required, and if it is not, the blinded token is signed and becomes available to the user. Since the token serves as evidence that she has not been blacklisted, no additional verification regarding the blacklist is necessary; and since tokens are unlinkable to each other or the user, privacy is preserved without need of a trusted third party. Faust thus provides efficient, TTP-free abuse prevention with per-user authentication costs that scale independently of the number of users or misusers of the SP.

2. RELATED WORK

Blind Signatures and Unlinkable Serial Transactions. Faust is heavily influenced by Unlinkable Serial Transactions (USTs) [19], which in turn use Chaum's blind signature scheme [5]. In this scheme, the signer has public key N, an RSA modulus, and secret key $d = 3^{-1} \bmod \phi(N)$.[1] We utilize a cryptographic hash function $H : \mathcal{M} \to \mathbb{Z}_N^*$,

[1]We note that any exponent e coprime to $\phi(N)$ could be used in place of 3, but since in our case it results in more

WPES'11, October 17, 2011, Chicago, Illinois, USA.

modeled as a random oracle. When a user wishes to obtain a blinded signature on the message $x \in \mathcal{M}$, she picks $r \in_R \mathbb{Z}_N^*$, and hands $\beta = H(x)r^3 \bmod N$ to the signer, who returns $\zeta = \beta^{1/3} \bmod N = H(x)^{1/3} r \bmod N$. Finally, the user computes $\sigma = \zeta / r = H(x)^{1/3} \bmod N$. It is easy to see that signing transcripts (β, ζ) are information-theoretically unlinkable to the signatures $(x, H(x)^{1/3} \bmod N)$; Bellare *et al.* [2] prove that creating $n+1$ valid signatures from n signing queries reduces to solving the *one-more RSA inversion problem*.

USTs utilize blind signatures to enforce anonymous, serial access to a subscription service. For example, imagine an online movie streaming service for which users pay a subscription fee, and suppose that each user wants to keep private which movies she watched. In the UST scheme. users pay a subscription fee and receive a signed token. Then each time the users wants to download a movie he spends his token and immediately receives a new blinded token. Each movie he watches is unlinkable to the other movies he watched because the tokens are all statistically independent of each other. Faust combines this idea with a method to privately and asynchronously retrieve the next token.

Nymble, BLAC, PEREA. Henry and Goldberg [10] provide an excellent overview of the TTP-based approaches to anonymous blacklisting [14, 24, 11, 15, 16, 9]; the most efficient of these have essentially no computational cost for the user and require only a handful of symmetric-key primitive operations from the SP, but leave the user and the SP vulnerable to collusion among the trusted parties. We note that most of these schemes explicitly rate-limit users to one authentication token per "linkability window," a period suggested to be 5 minutes.

BLAC [21] completely eliminates the dependence on TTPs as follows: each user receives an anonymous credential (a blinded signature on the value g^x in an appropriate group). When the user authenticates, he picks a random base h, and sends the SP the pair (h, h^x), along with a proof that he knows a signature on g^x. If the user misbehaves, the value (h, h^x) will be added to the blacklist; so to complete the authentication the user also proves that for each (b, c) on the blacklist, $b^x \neq c$. This means that in BLAC, each authentication requires computation and bandwidth linear in the number of blacklisted users. Two similar schemes [4, 3] appeared independently in the same year.

To reduce the computational and communication complexity of these TTP-free schemes, PEREA [22] modifies the blacklisting requirement: In PEREA, the user produces a one-time random token for each authentication and proves that none of her previous K tokens was blacklisted. Thus, the "window" during which she can be effectively blacklisted for a given action is limited. Our approach has similar blacklisting semantics, but because we eliminate the explicit blacklist, there is no dependence on a "window size" K in Faust, whereas authentication cost in PEREA has a linear dependence on K.

3. FAUST

We present Faust, a solution to the anonymous credential problem with blacklisting. It is efficient and does not use third parties, but it does change the blacklisting semantics slightly, requiring that the SP judge within a fixed time whether a particular action was acceptable or not.

Faust is based on digital tokens. Each user Alice receives a token (or K tokens) during registration. Every time she wants to interact with the SP, she connects through an anonymizing network, spends a token, and completes an action like posting content to a page. To allow the SP to send her a new token in the future, she also provides a blinded token. If her action is not flagged as inappropriate within a time period of Δ seconds, say an hour, the service provider will sign her token. When Alice sees her post was approved, she retrieves the blinded token and un-blinds it to restore her editing power. Only Alice knows the unblinding factor, so only she can unblind the token.

Blacklisting in Faust has different timing constraints than blacklisting in Nymble or BLAC. In Faust, once the SP signs a blinded token for Alice, it can never blacklist her for that action, whereas in Nymble or BLAC the service provider can blacklist a user for an action long after it occurred. On the other hand, in BLAC and Nymble a user committing vandalism can keep committing vandalism until the first instance is detected. To limit misbehavior, both systems rate-limit users to one action every time period, say every five minutes. If this period is short, then users can misbehave several times before they are noticed, while if this period is long then even legitimate users are slowed. In Faust, the SP always has time to judge a user's current action before the user receives a new token, so the user cannot misbehave twice with the same token.

3.1 Protocol

User Registration. To begin using Faust, each user needs to receive a token. Since having a valid, unused token (x, $H(x)^{1/3} \bmod N$) "whitelists" a user, it is important to limit the number of tokens any user can receive. As discussed by Henry and Goldberg [10], this could be done based on several scarce resources. Here we outline two approaches.

IP address. To limit misuse based on IP addresses, the client registers by connecting directly (not anonymously) to either the SP or a third party Token Manager (TM, trusted by the SP) and requesting a token. The client sends a blinded token; the TM checks that no token has been previously issued to the requesting IP address, and (if not) signs the token and returns it to the client, who unblinds it. If a third party is used for registration, the user then sends the token along with another blinded token (using the SP's public key) to the SP to obtain a token signed under the SP's public key.

Since IP-addresses may be reassigned, the TM should periodically clear its list of IP addresses, breaking time into windows of say, one week. To prevent users from saving these IP based tokens over time, the SP can choose a new public key for each window. This will effectively "forgive" users who misbehaved during the previous window, as in Nymble. We note that the use of IP addresses as a scarce resource, present in the literature on anonymous blacklisting since Nym [13], has known disadvantages due to the dynamicity of IP addresses. However, as Johnson *et al.* note [14], many SPs currently employ IP-based blacklisting to successfully block the worst abusers, so using IP addresses in anonymous blacklisting schemes would give SPs the same level of

efficient verification with no known disadvantages, we focus on the case $e = 3$

protection against anonymous users as they enjoy against non-anonymous misusers.

Currency. Users who cannot directly connect to the SP (without use of an anonymity network) could be accommodated using an electronic payment mechanism (such as EFT or PayPal). In this case, the user connects anonymously to the SP, and submits payment information along with a blinded token. The SP processes the payment and signs the token. Notice that the payment mechanism need not be anonymous, because the SP will be information-theoretically unable to link the unblinded token with the payment.

Authentication. When a user Alice would like to authenticate to the SP, she does the following:

1. She connects to the SP through an anonymity network.
2. Alice downloads the SP's current public key N. She prepares a blinded token for signing by choosing a random message $x \in_R \mathcal{M}$ and a blinding factor $r \in_R \mathbb{Z}_N^*$ and then computing a blinded value $\beta = yr^3 \bmod N$.
3. Alice uploads β, a previously obtained token $(x', \alpha' = H(x')^{1/3})$, and the content of her post.
4. The SP verifies that her token is valid by ensuring that $\alpha'^3 = H(x')$ and verifies that it has never been spent, and then stores β in a database along with her content. There is a public, unique id number i associated with each post.

Token Retrieval. After Δ seconds have passed, where Δ is a public parameter which gives the SP a chance to detect abuse, the SP checks whether an action was acceptable or not. When a post with id i and blinded value β is declared acceptable by the SP, it signs the blinded value to get $\alpha = \beta^{1/3}$ and adds the pair (i, α) to a list A of signatures for accepted posts.

Notice that if Alice waits at least Δ seconds, and then requests the token associated with post i^* in order to make her next post with id j^*, there will be a traffic analysis attack in which the SP can infer that post i^* and j^* are likely linked because of the proximity between the request for token i^* and the appearance of post j^*. Thus, some scheme is needed that obscures which tokens have been requested.

In the most straightforward scheme, after Alice has waited Δ seconds for one of her posts i^* to be judged, she connects to the SP through an anonymity network and downloads the list of accepted posts $A = \{(i, \alpha_i)\}$. If i^* is in the list A, Alice unblinds the signature $\alpha = \alpha_{i^*}$ by computing $\sigma = \alpha/r = H(x)^{1/3} \bmod N$ and stores it for future use.

Unfortunately, the length of the list A of signatures for accepted posts grows in proportion to the total number of posts made, so downloading A during the token retrieval protocol would become expensive. One option to address this would be to use an efficient Private Information Retrieval [20, 17] scheme, possibly offloaded to a third party; however, this would still incur significant costs.

Faust employs several mechanisms to reduce the cost of privately retrieving tokens. First, time is partitioned into periods of length $\ell \geq \Delta$; each user retrieves her token from the list A of blinded tokens associated with the period in which she last authenticated. The period length ℓ is chosen so that each Faustian user of the SP is likely to access the SP at least once within each period.

Second, the user does not retrieve the entire list of tokens A from the period. Instead, she requests an expected fraction $1/M$ of the of the entries of A. In this variation, during the token retrieval protocol for post i^*, the user sends $i^* \pmod M$ to the server. The server "filters" the list A to get $A' = \{(i, \alpha) \in A : i \equiv i^* \pmod M\}$ and returns A' from which the user can extract (i^*, α_{i^*}). (Note that the server can easily store the list A in M buckets so that no extra computation is required)

Finally, after retrieving her token, the user chooses a random delay τ uniformly from the interval $[0, T]$, (where T is a public parameter of the SP) and waits at least τ seconds before using the token. The parameters T and M are chosen so that with high probability, every $i \in \{0, \ldots, M\}$ will be requested at least once within T seconds. (For example, if the arrival rate of Faust users to the SP is λ, we can choose T, M to satisfy $T > 2M/\lambda$.) Since the unblinded tokens are unlinkable to the blinded tokens, this mixes the user's token with all tokens from the previous period. An impatient or security-conscious user can send the special request $M + 1$, downloading all tokens from the period, and have the token available for immediate use while ensuring his privacy.

In general, the parameters T, M, and ℓ can be adjusted to control the number of times any token is retrieved, at the expense of a longer wait between user actions, and dependent only on the arrival rate of Faustian users to the SP.

4. EVALUATION

4.1 Security

We briefly sketch how our construction achieves the security goals identified in [24] — *Blacklistability*, *Rate-limiting*, *Non-frame-ability*, and *Anonymity* — assuming the *one-more RSA inversion problem* [2] is computationally intractable.

Blacklistability. An honest Token Manager will only issue 1 token per user. Thus for a coalition of c users to authenticate after all have been blacklisted, they would have to forge a token, violating the assumed intractability of the *one-more RSA inversion problem*.

Non-Frameability. When a user Alice chooses a value $y = H(x)$, she only ever transmits $r^3 y$ which is statistically independent of y. If another user Bob downloads her blinded token $ry^{1/3} \bmod N$, all he has is a pair (z, z^3) where $z = ry^{1/3} \in_R \mathbb{Z}_N^\times$. He could produce such a pair on his own, and it is no help to him in forging any token, including Alice's token. Thus if the SP publishes a signature on Alice's blinded token, only Alice can use it to create a token, and no other user can spend her token without simply guessing it.

Anonymity. When a user Alice connects to the SP to post content, she only identifies herself by submitting a token σ. This token is only used once and statistically independent of all her previous interactions. Thus the SP has no information with which to identify Alice or link her posts together.

4.2 Efficiency

Faust is much more efficient than previous TTP-free solutions. There are no interactive proofs. The only computation for each authentication is for the SP to sign a token,

Blacklisting Time (Δ)	140 minutes
Token list period (ℓ)	16 hours
Client arrival rate (λ)	6.3 / minute
Authentications per period ($\ell \times \lambda$)	6000
List reduction factor (M)	60
Token retrieval bandwidth	25 KiB
Maximum mixing interval (T)	20 minutes

Table 1: Suggested parameters for Wikipedia deployment, if all anonymous edits require Faust. The resulting deployment would require 25KiB of bandwidth (on average) per authentication, the same as the average request served by Wikipedia; and the expected time between authentications would be $\Delta + T/2 = 150$ minutes.

the user to blind and unblind a token, and the SP to verify a token. Each of these steps can be done with a single modular exponentiation (requiring 3.3ms at 2048 bits on a 2.67GHz Intel Xeon W3520) in Chaum's scheme. The total bandwidth required by both parties to an authentication is the length of one $x \in \mathcal{M}$, and $3 + c$ values in $\mathbb{Z}_N^*$, where c is the constant number of times each blinded token is downloaded. (We expect a typical value of c to be on the order of 10-100, but the best value will depend on the SP.)

In contrast, PEREA requires at least $(K+1)D + 6K + 3$ modular exponentiations by the client (where D is the average number of updates to the blacklist between two authentications by the client) and $\frac{14K}{3}+3$ modular exponentiations by the server [22]. The bandwidth required per authentication is $D + \frac{14K}{3} + 3$ elements of $\mathbb{Z}_N^*$. Thus at the same security level, Faust is roughly an order of magnitude faster than PEREA with $K = 1$.

4.3 Wikipedia example

Faust has several timing parameters that rely on specific properties of the service it protects. Here we give an example of concrete settings based on measurements from Wikipedia. These settings are summarized in Table 1.

Blacklisting time Δ. To investigate the proper value of Δ – the time in which an abuse must be noticed – we collected data on Wikipedia reversions. We developed a script which scans the revision history of an article from Wikipedia for the keywords "revert" or "rv" – which users often use when reverting vandalism – and notes the time elapsed between editing and reversion. We then used Wikipedia's "Special:Random" page to find random pages until we had found 1000 reverts. The resulting cumulative distribution is shown in Figure 1.

The median reversion time in this data set is 6 minutes, and the 75th percentile is 140 minutes. It is likely that the worst incidents of vandalism are noticed quickly, so the reverts which occurred after an hour may not merit blacklisting. We also note that a large study of reverts on Wikipedia [18] found that 89% of vandalism instances were repaired within 100 views. In December 2010, each of the 1000 most popular pages were viewed at least 5 times per minute.[2]

[2]See "Wikipedia article traffic statistics", `http://stats.grok.se/en/top`

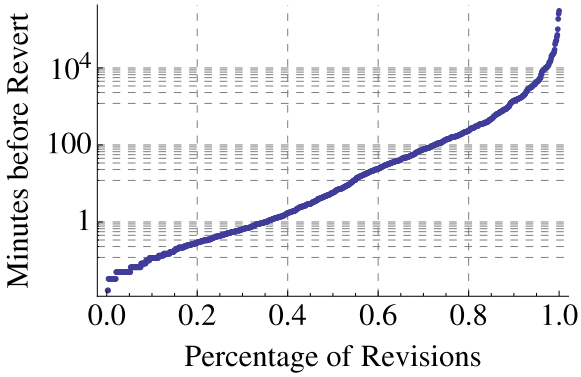

Figure 1: Plot of the number of minutes before a revert versus the percentage of reverts occurring before that time for random pages on Wikipedia.

Thus, if users can be directed to articles with pending Faust tokens, Δ values as low as 30 minutes seem feasible.

Arrival Rate λ and Period length ℓ. We collected the complete list of 3666 Wikipedia edits for a 30-minute period on July 15, 2011. This list contains 813 edits that were made by users with no login; among these users, 188 had a previous edit session: a chain of edits $e_1, e_2, \ldots, e_k$ such that the time between each e_i and e_{i+1} is less than 5 minutes. We use these pseudonymous users as our model for Faustian Wikipedians. The arrival rate of of these users was $\lambda = 188/30 \approx 6.3$ per minute. We then retrieved the editing history of each of these users (up to 500 edits) and computed the time between editing sessions. The median time between the 10628 edits by these users was 14.4 hours, suggesting a period of $\ell = 16$ hours. At an arrival rate of 6.3 per minute, we expect roughly 6000 tokens on each period's token list. Downloading the list would consume 1.5 MiB at 2048 bits per token.

Client wait T for reduction M. The average request served by Wikipedia results in approximately 25 KiB of data;[3] thus to reduce the size of the token retrieval message to that of an average page view, it is sufficient to take $M = 60$. With $M = 60$, this gives a wait time $T = 2 \times 60/6.3 \approx 20$ minutes.

5. VARIATIONS

Multiple Tokens. An obvious limitation of Faust as described thus far is the fact that a user can only use the service once every $\Delta + T/2$ seconds, on average. If this time is close to 150 minutes as computed in the previous section, this represents a serious usability constraint. A natural approach to resolve this issue is to give each user some number $K > 1$ of tokens, similar to PEREA. This decreases the average time between uses by a factor of K, and allows "bursts" of up to K uses in short succession. Because all tokens are unlinkable, there is no change in the privacy provided by the system, and the cost per authentication remains the same.

Once a user has multiple tokens, the semantics of blacklisting become similar to the "d-strikes-out" policy of [23, 1]: a user who misbehaves less than K times can still use the service; once the user is caught misbehaving K times he becomes unable to access the service. Unlike the BLAC/PEREA

[3]`http://en.wikipedia.org/wiki/Wikipedia:Statistics` shows 48K requests/s and 9.5Gbps for July 2011.

d-strikes out policy, however, a Faust user that misbehaves fewer than K times will incur a penalty, in the form of an increased mean time between authentications.

Naughtiness. Recently, Au *et al.* [1] have introduced the notion of "naughtiness" in PEREA: each misbehavior by the user is assigned a severity, and the user is allowed to authenticate as long as the severity is below some threshold η. We note that a similar type of policy can be supported by multi-token Faust: let ζ be the maximum severity assigned to an action. Then each user receives $\eta + \zeta - 1$ tokens on registration. Each time the user authenticates, she submits ζ valid tokens along with ζ blinded tokens. If the SP determines within time Δ that the user misbehaved with severity s, then the SP signs and returns only $\zeta - s$ of the blinded tokens. As long as the total severity of a user's misbehavior is less than η, she can continue to use the service (at a reduced rate), and once her total severity exceeds η she will no longer have the ζ valid tokens necessary to authenticate. We note however, that this solution increases the cost of authentication by a factor of ζ.

Forgiving tokens. One feature of BLAC and PEREA is the ability of the SP to forgive a user and remove her from the blacklist. This aligns with current Wikipedia IP blocking policy which usually blocks IP addresses for a limited length of time. Limited time blocks are also possible in single-token Faust: the service provider can simply delay signing the new token until the block expires.

Recovering tokens. In the basic system, users must store the tokens they receive and the blinding factors for pending tokens on some medium, and if that information is lost the user must either wait until the end of the current token window (if using IP-based token limiting) or pay for a new token (if using currency-based token limiting). However, if the user can remember or write down a single high-entropy password generated by the Faust software, then it is possible for the user to generate blinding factors pseudo-randomly, so the user can recover all of her tokens from that password. In this modification, a cryptographic pseudo-random function F is used, and users generate their blinding factors using their passwords.

When a user Alice with password sk is ready to post an edit, she asks the SP for the index i of the next post and then computes her blinding factor and message $(r, x) = F_{sk}(i)$ which she then uses as in the standard protocol. Later if her edit was approved, she can retrieve that token (or re-retrieve it) even if she is on a different computer. She can find one of her posts, download i and the blinded signature, compute the blinding factor from sk and i, and then unblind the token.

Acknowledgments

We thank Roger Dingledine, Ian Goldberg, Ryan Henry, Yongdae Kim and Zi Lin for helpful discussions and comments about the anonymous blacklisting problem in general, as well as the FAUST protocol in particular. This work was partially supported by a University of Minnesota Undergraduate Research Opportunities Program (UROP) grant to Peter Lofgren, and the National Science Foundation under grant 0546162.

6. REFERENCES

[1] M. H. Au, P. P. Tsang, and A. Kapadia. PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. Technical Report TR688, Indiana University, 2011.

[2] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The One-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.

[3] S. Brands, L. Demuynck, and B. D. Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415. Springer, 2007.

[4] E. Brickell and J. Li. Enhanced Privacy ID: a Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 21–30, New York, NY, USA, 2007. ACM.

[5] D. Chaum. Security without identification: transaction systems to make Big Brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[6] U. I. Corp. Ultrasurf: Privacy. security. freedom. `http://www.ultrareach.com/`, July 2011.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[8] J. GmbH. JonDoNym: Private and secure web surfing. `http://anonymous-proxy-servers.net/`, September 2010.

[9] R. Henry and I. Goldberg. Extending Nymble-like systems. In *Proc. 32nd IEEE Symposium on Security & Privacy*, 2011.

[10] R. Henry and I. Goldberg. Formalizing anonymous blacklisting systems. In *Proc. 32nd IEEE Symposium on Security & Privacy*, 2011.

[11] R. Henry, K. Henry, and I. Goldberg. Making a nymbler Nymble using VERBS. Technical report, University of Waterloo Technical Report CACR 2010-05, 2010. Extends [12].

[12] R. Henry, K. Henry, and I. Goldberg. Making a nymbler Nymble using VERBS. In *PETS: Proceedings of the 10th Privacy Enhancing Technologies Symposium*. Springer, 2010.

[13] J. Holt and K. Seamons. Nym: Practical pseudonymity for anonymous networks. *BYU Internet Security Research Lab Technical Report*, 4, 2006.

[14] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith. Nymble: Anonymous IP-address blocking. In *Proceedings of The Seventh International Symposium on Privacy Enhancing Technologies (PET), Ottawa, Canada*, volume 4776 of *LNCS*, pages 113–133. Springer-Verlag, June 2007.

[15] Z. Lin and N. Hopper. Jack: Scalable accumulator-based Nymble system. In *WPES2010: Proceedings of the 9th ACM Workshop on Privacy in the Electronic Society*. ACM, 2010.

[16] P. Lofgren and N. Hopper. BNymble (a short paper): More anonymous blacklisting at almost no cost. In *Fifteenth International Conference on Financial Cryptography and Data Security*. Springer, 2011.

[17] F. Olumofin and I. Goldberg. Revisiting the computational practicality of Private Information Retrieval. In *Fifteenth International Conference on Financial Cryptography and Data Security*. Springer, 2011.

[18] R. Priedhorsky, J. Chen, S. Lam, K. Panciera, L. Terveen, and J. Riedl. Creating, destroying, and restoring value in Wikipedia. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 259–268. ACM, 2007.

[19] S. Stubblebine, P. Syverson, and D. Goldschlag. Unlinkable serial transactions: protocols and applications. *ACM Transactions on Information and System Security (TISSEC)*, 2(4):354–389, 1999.

[20] J. T. Trostle and A. Parrish. Efficient computationally private information retrieval from anonymity or trapdoor groups. In M. Burmester, G. Tsudik, S. S. Magliveras, and I. Ilic, editors, *ISC10: Proceedings of the Information Security Conference*, volume 6531 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2010.

[21] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable Anonymous Credentials: blocking misbehaving users without TTPs. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81, New York, NY, USA, 2007. ACM.

[22] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. PEREA: Towards practical TTP-free revocation in anonymous authentication. In *CCS '08: Proceedings of the 14th ACM conference on Computer and communications security*, pages 333–344. ACM, 2008.

[23] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. *ACM Trans. Inf. Syst. Secur.*, 13:39:1–39:33, December 2010.

[24] P. P. Tsang, A. Kapadia, C. Cornelius, , and S. W. Smith. Nymble: Blocking Misbehaving Users in Anonymizing Networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, Sept. 2009.