TrollThrottle — Raising the Cost of Astroturfing

Ilkan Esiyok[1], Lucjan Hanzlik[2], Robert Künnemann[1], Lena Marie Budde[3], and Michael Backes[1]

[1] CISPA Helmholtz Center for Information Security
[2] CISPA Helmholtz Center for Information Security, Stanford University
[3] Saarland University

Abstract. Astroturfing, i.e., the fabrication of public discourse by private or state-controlled sponsors via the creation of fake online accounts, has become incredibly widespread in recent years. It gives a disproportionally strong voice to wealthy and technology-savvy actors, permits targeted attacks on public forums and could in the long run harm the trust users have in the internet as a communication platform.
Countering these efforts without deanonymising the participants has not yet proven effective; however, we can raise the cost of astroturfing. Following the principle 'one person, one voice', we introduce TrollThrottle, a protocol that limits the number of comments a single person can post on participating websites. Using direct anonymous attestation and a public ledger, the user is free to choose any nickname, but the number of comments is aggregated over all posts on all websites, no matter which nickname was used. We demonstrate the deployability of TrollThrottle by retrofitting it to the popular news aggregator website Reddit and by evaluating the cost of deployment for the scenario of a national newspaper (168k comments per day), an international newspaper (268k c/d) and Reddit itself (4.9M c/d).

1 Introduction

Astroturfing describes the practice of masking the sponsor of a message in order to give it the credibility of a message that originates from 'grassroots' participants (hence the name). Classic astroturfing involves paid agents fabricating false public opinion surroundings, e.g., some product. The anonymity of the cyberspace makes astroturfing very inexpensive; now, it can even be mechanised [22]. This form of astroturfing, also called 'cyberturfing', is a Sybil attack that exploits a useful, but sometimes fallible heuristic strategy in human cognition: roughly speaking, the more people claim something, the improved judgement of credibility [32, 33]. In the wake of the 2016 US elections, Twitter identified, '3,814 [..] accounts' that could be linked to the Internet Research Agency (IRA), a purported Russian 'troll factory'. These accounts 'posted 175,993 Tweets, approximately 8.4% of which were election-related' [44], which is likely only a fraction of the overall activity. This influence comes at a modest price, as the IRA had a $1.25M budget in the run-up to the 2016 presidential election [3] and only 90 members of staff producing comments [17].

The everyday political discourse has also suffered. Many newspapers have succumbed under the weight of moderation, e.g., the New York Times [21]. Some newspapers decided to move discussion to social media [48], where they only moderate a couple of stories each day and leave out sensitive topics such as migration altogether [39]. Kumar et. al. show that many popular news pages have hundreds of active sock puppets, i.e., accounts controlled by individuals with at least one other account [31]. The New York Times, one of the largest newspapers worldwide, has put serious effort and technological skills into moderating discussion, but ultimately, they had to give up. In mid-2017, they reported how they employ modern text analysis techniques to cluster similar comments and moderate them in one go. At that point in time, they had 12 members of staff dedicated to moderation, handling a daily average of 12,000 comments [35]. Despite the effort and expertise put into this, they had to give up three months later, deactivating the commenting function on controversial topics [21].

In this paper, we propose a cryptographic protocol that permits throttling the number of comments that a single user can post on all participating websites *in total*. The goal is raising the cost of astroturfing: if the threshold is τ, the cost of posting n comments is the cost of acquiring $\lceil \frac{n}{\tau} \rceil$ identities, be it by employing personnel, by bribery or by identity theft. Our proposal retains the anonymity of users and provides accountability for censorship, i.e., if a user believes her comment ought to appear on the website, she can provide evidence that can be evaluated by the public to confirm misbehaviour on the part of the website. Part of this system is a pseudo-random audit process to ensure honest behaviour, which we have formally verified.

We show that this protocol, TrollThrottle, can be retrofitted to existing websites. We set up a forum[4] on Reddit that demonstrates our proposal. We also compute the additional cost of operation incurred by our protocol by simulating user interaction for three real-life scenarios: an international newspaper, a nationwide publication and all comments posted on Reddit in one day. In the newspaper case, the computational overhead incurs a cost of about $1.20; for the whole of Reddit, $3.60 is sufficient.

As a by-product and second contribution, we extend the notion of direct anonymous attestation (DAA) by proposing two features with applications outside our protocol. Both are already supported by an existing DAA scheme by Brickell and Li [14]. First, updatability, which means that the issuer can non-interactively update the users' credentials. This allows for easy key rollover in the mobile setting and for implicit revocation of credentials by not updating them (old credentials invalidated). Second, instant linkability, which means that each signature contains a message-independent pseudonym that determines whether two signatures can be linked. This allows to efficiently determine whether a signature can be linked to any existing signature within a given set.

[4] https://old.reddit.com/r/trollthrottle/

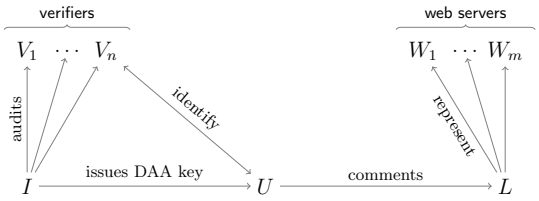

Fig. 1: Approach

2 TrollThrottle

Despite text analysis techniques that can facilitate moderation, e.g., clustering [35], many local and international newspaper websites gave up on moderating and disabled commenting sections [21, 48]. Even if troll detection could be automated, e.g., via machine learning, as soon as the detection algorithm becomes available to attackers, numerous techniques permit the creation of adversarial examples [34] to evade classifiers. Fundamentally, astroturfing does not even rely on automated content generation and can be conducted by paid authors in countries with low labour cost: e.g., the so-called 50-cent party, a group of propagandists sponsored by China, was named after the remuneration they receive per comment [36].

Our approach is orthogonal to detection by content. If we can limit the number of messages to a certain threshold τ that each physical person can send per day, bots become largely useless, and troll farms need to pay, bribe or steal identities from sufficiently many actual people to send messages in their name. Besides raising the cost, this also raises the probability of detecting larger operations.

We built our approach on direct anonymous attestation (DAA [12]). In DAA, an issuing party distributes membership credentials to signers (in our case users) that it considers legitimate. Each signer can prove membership by signing data: a valid DAA signature guarantees that a valid signer signed this data, but does not reveal the signer's identity. DAA schemes can also be seen as group signature schemes that prevent the issuing party to identify the signer of the message, a feature known as *opening*.

To avoid a single point of trust, the identification of the user is not only a matter of the issuer, who is likely to be the provider of this service. Instead, an agreed upon set of verifiers establishes the legitimacy of users, i.e., that they are real people and that they have not received a DAA key before. We will discuss how the issuer and the verifiers keep each other honest in the full version [8]. To provide accountability, a public ledger keeps records about the comments that websites ought to publish.

Thus, the following parties cooperate in TrollThrottle: an issuer I, who issues DAA keys, a set of verifiers V, who verify the users' identities, a set of users U, who create DAA signatures of their comments, a public append-only ledger L,

who records these signatures, and a set of websites, who verify these signatures and are bound to publish comments whose signatures exist on the ledger.

In DAA, a signature can be created and verified with respect to a so-called *basename*. Signatures created by the same user with the same basename can be linked. This is the key feature to achieve throttling. Within a commenting period t, e.g., a day, only signatures with a basename of the form (t, seq) are accepted, where seq is a sequence number between 1 and the desired threshold τ. If a user signs two messages with the same basename, they can be linked and discarded by the website. Hence a user can create at most τ signatures that are unlinkable to each other. A valid DAA signature assures the website that a valid user signed this comment, but neither the website, nor the issuer or the verifier learns who created the comment, or which other comments they created.

By storing the signatures on the ledger L, the websites (a) can enforce a global bound, and (b) provide accountability for censorship by promising to represent all comments addressed to this website that appear in the ledger. If a website does not publish a user's comment, it must have sufficient grounds for censorship.

We build on Brickell and Li's DAA scheme [14] for its efficiency, but extend it with various features to make TrollThrottle more efficient (see the full version [8, Appendix A]), more secure (Section 3), more practical (Section 4), and more resistant against compromise (Section 4).

We assume the issuer I is known to all users and websites; the verifiers V are known to the issuer and all websites; and the public ledger L is known to all participants in the protocol. The ledger can be implemented using a consensus mechanism between the websites and some trusted representatives of civil society (e.g., via Tendermint [24] or PBFT [18]) or open consensus mechanisms like blockchains. We will formalise our approach in terms of PPT algorithms and an interactive protocol.

Definition 1 (Accountable commenting scheme). *An accountable commenting scheme consists of a tuple of algorithms* (Setup, KeyGen, $\mathsf{Comment}$, Verify, Claim, $\mathsf{VerifyClaim}$) *and an interactive protocol* ($\mathsf{Join} - \mathsf{Issue}$). *The algorithms and the protocol are specified as follows.*

$\mathsf{Setup}(1^\lambda)$ models the generation of a setup parameter ρ used by all participants from the security parameter 1^λ. This parameter is an implicit argument for the other algorithms, but we omit it for brevity. The issuer I invokes $\mathsf{KeyGen}(\rho)$ to generate its secret key sk_I and public key pk_I from this parameter.

The issuing procedure $\langle \mathsf{Join}(pk_\mathrm{I}, U) \leftrightarrow \mathsf{Issue}(sk_\mathrm{I}, \mathsf{ver}, U)\rangle$ is an interactive protocol between I and a new user (identified with U) that has not registered so far. At the end of the protocol, the user receives a credential $cred_U$ and a secret key sk_U. For now, we abstract away from the verifiers by giving the issuer access to a read-only database ver such that $\mathsf{ver}[V, U] \in \{0, 1\}$ is 1 iff the verifier V confirms the identity of a user. In Section 4, we present and verify a protocol to implement and audit this verification step.

The commenting procedure is split into four PPT algorithms, $\mathsf{Comment}$ for U to generate comments that she sends to the ledger, Verify for W to verify that

a comment on the ledger should be displayed, Claim for U to generate a claim that a valid comment on the ledger ought to be published, and $\mathsf{VerifyClaim}$ for the public to verify that said claim is valid.

$\mathsf{Comment}(pk_{\mathrm{I}}, sk_U, cred_U, dom, m)$ is executed by U, who knows the issuer's public key pk_{I}, its own secret key sk_U and credentials $cred_U$. U chooses a basename $dom \in \{0,1\}^*$ and a message $m \in \{0,1\}^*$ and obtains a signed comment γ and a pseudonym nym, both of which she stores on the ledger. The basename determines a user's nym, so that anyone can check whether two comments were submitted with the same basename by checking their respective nyms for equality. This is a key feature: in TrollThrottle, all basenames have to be of the form $\langle t, i\rangle$ for a commenting period t and an integer $i \in \{1, \ldots, \tau\}$. Hence there are at most τ unique basenames within t, and thus at most τ nyms per sk_U and t.

$\mathsf{Verify}(pk_{\mathrm{I}}, nym, dom, m, \gamma)$ can be computed by any website that has access to the issuer's public key pk_{I}, the comment on the ledger γ, pseudonym nym, domain dom (which can be determined by trial and error) and a message $m \in \{0,1\}^*$ received from the user. If the output is 1 and γ is valid w.r.t. m, the website W must display m. If W fails to do that, the user computes $\mathsf{Claim}(pk_{\mathrm{I}}, sk_U, cred_U, dom, m, \gamma)$ on the same data as before. The output *evidence* can be publicly verified using the $\mathsf{VerifyClaim}(pk_{\mathrm{I}}, dom, m, \gamma, evidence)$ algorithm. It outputs 1 iff *evidence* and the ledger entry γ prove that m ought to be displayed during the commenting period indicated by dom.

3 Protocol definition

Before we present TrollThrottle as an instance of an accountable commenting scheme, we introduce the necessary cryptographic notions.

We follow the DAA definition proposed in [14]. A DAA scheme consists of four PPT algorithms $(\mathsf{Setup}_{\mathsf{DAA}}, \mathsf{Sign}_{\mathsf{DAA}}, \mathsf{Verify}_{\mathsf{DAA}}, \mathsf{Link}_{\mathsf{DAA}})$ and an interactive protocol $(\mathsf{Join} - \mathsf{Issue}_{\mathsf{DAA}})$, between parties: an issuer I, a verifier V and a signer S. In our case, the websites take the role of the verifiers, and the users the role of the signer.

$\mathsf{Setup}_{\mathsf{DAA}}(1^\lambda)$ is run by I; based on the security parameter 1^λ, it computes the issuer's secret key sk_I and public key pk_I, including global public parameters. $\mathsf{Join} - \mathsf{Issue}_{\mathsf{DAA}}$ is an interactive protocol between I and S to provide credentials issued by I to S. It consists of sub-algorithms $\mathsf{Join}_{\mathsf{DAA}}$ and $\mathsf{Issue}_{\mathsf{DAA}}$. S executes $\mathsf{Join}_{\mathsf{DAA}}(pk_I, sk_S)$ on input pk_I and sk_S to obtain the commitment com.[5] I executes $\mathsf{Issue}_{\mathsf{DAA}}(sk_I, \mathsf{com})$ to create a credential $cred_S$ that is associated with sk_S and sent to S. Note the key of S remains hidden from I.

$\mathsf{Sign}_{\mathsf{DAA}}(sk_S, cred_S, dom, m)$ is executed by S to create a signature σ for a message m w.r.t. a basename dom, which is optionally provided by V. If $dom \neq \perp$, signatures created by the same signer can be linked.

$\mathsf{Verify}_{\mathsf{DAA}}(pk_I, m, dom, \sigma, RL)$ is a deterministic algorithm run by V on a message m, a basename dom, a signature σ, and a revocation list RL to determine

[5] We slightly alter the original definition and assume that instead of sampling this key inside the algorithm, S provides the key as an input.

if a signature σ is valid. In [14], I stores revoked secret keys in the revocation list RL; signatures created with a revoked secret key are not valid.

$\mathsf{Link}_{\mathsf{DAA}}(\sigma_0, \sigma_1)$ is a deterministic algorithm that determines with overwhelming probability whether signatures σ_0 and σ_1 were created by the same signer with the same basename $dom \neq \perp$. It outputs 1 if the signatures are linked, 0 for unlinked and $\perp$ for invalid ones.

DAA features Brickell and Li's DAA scheme [14] has the following security properties (formally stated in the full version [8, Appendix F]).

Correctness: if an honest signer's secret key is not in the revocation list RL, then, with overwhelming probability, signatures created by the signer are accepted and correctly linked by an honest verifier.

User-controlled-anonymity: a PPT adversary has a negligible advantage over guessing in a game where she has to distinguish whether two given signatures associated with different basenames were created by the same signer or two different signers.

User-controlled-traceability: no PPT adversary can forge a non-traceable yet valid signature with $dom \neq \perp$[6] without knowing the secret key that was used to create the signature, or if her key is in the revocation list RL.

We add the following property (formally stated in the full version [8, Appendix A]):

Instant-linkability There is a deterministic poly-time algorithm NymGen s.t. $\mathsf{NymGen}(sk_S, dom)$ generates a nym that is otherwise contained in the signature, and two nyms are equal iff the corresponding signatures are linkable.

Zero-knowledge The user creates non-interactive proofs of knowledge to show that her key was honestly generated. We highlight the notation here and refer to the full version [8, Appendix C.1] for the full security definitions. Let $\mathcal{R}$ be an efficiently computable binary relation. For $(x, w) \in \mathcal{R}$, we call x a *statement* and w a *witness*. Moreover, $L_{\mathcal{R}}$ denotes the language consisting of statements in $\mathcal{R}$, i.e., $L_{\mathcal{R}} = \{x \mid \exists w : (x, w) \in \mathcal{R}\}$.

Definition 2. *A non-interactive proof of knowledge system Π consists of the following three algorithms* $(\mathsf{Setup}, \mathsf{CreateProof}, \mathsf{VerifyProof})$. $\mathsf{Setup}(1^\lambda)$**:** *on input security parameter 1^λ, this algorithm outputs a common reference string ρ.* $\mathsf{CreateProof}(\rho, x, w)$**:** *on input common reference string ρ, statement x and witness w; this algorithm outputs a proof π.* $\mathsf{VerifyProof}(\rho, x, \pi)$**:** *on input common reference string ρ, statement x and proof π; this algorithm outputs either* 1 *or* 0.

TrollThrottle We will now present TrollThrottle in terms of an accountable commenting scheme (see Def. 1). Besides an instantly linkable DAA scheme, we assume a collision-resistant hash function h and a non-interactive proof of knowledge system for the relation:

[6] Note that basenames in TrollThrottle are always different from $\perp$, see Section 3.

$$((\mathsf{com}, pk_{I,\mathsf{DAA}}), (sk_{S,\mathsf{DAA}})) \in \mathcal{R}_{\mathsf{Join}} \iff \mathsf{com} \xleftarrow{\$} \mathsf{Join}_{\mathsf{DAA}}(pk_{I,\mathsf{DAA}}, sk_{S,\mathsf{DAA}}).$$

We assume that the witness for the statement $(\mathsf{com}, pk_{I,\mathsf{DAA}})$ contains the random coins used in $\mathsf{Join}_{\mathsf{DAA}}$.

Definition 3. *TrollThrottle Protocol*

$\mathsf{Setup}(1^\lambda)$ - *compute the parameters for the zero-knowledge proof of knowledge* $\rho_{\mathsf{Join}} \xleftarrow{\$} \mathsf{Setup}_{\mathsf{ZK}}(1^\lambda)$ *and output* $\rho = (1^\lambda, \rho_{\mathsf{Join}})$.

$\mathsf{KeyGen}(\rho)$ - *execute* $(pk_{I,\mathsf{DAA}}, sk_{I,\mathsf{DAA}}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{DAA}}(1^\lambda)$, *set and return* $pk_{\mathrm{I}} = pk_{I,\mathsf{DAA}}$ *and* $sk_{\mathrm{I}} = (pk_{I,\mathsf{DAA}}, sk_{I,\mathsf{DAA}})$.

$\mathsf{Join}(pk_{\mathrm{I}}, sk_U, U)$ - *let* $pk_{\mathrm{I}} = pk_{I,\mathsf{DAA}}$ *and* $sk_U = sk_{S,\mathsf{DAA}}$. *Run* $\mathsf{com} \xleftarrow{\$} \mathsf{Join}_{\mathsf{DAA}}(pk_{I,\mathsf{DAA}}, sk_{S,\mathsf{DAA}})$ *and compute proof* $\Pi_{\mathsf{Join}} = \mathsf{CreateProof}(\rho_{\mathsf{Join}}, (\mathsf{com}, pk_{I,\mathsf{DAA}}), sk_{S,\mathsf{DAA}})$. *Send* $(\mathsf{com}, \Pi_{\mathsf{Join}})$ *to the issuer and receive* $cred_U$. *Return* $(cred_U, sk_U)$.

$\mathsf{Issue}(sk_{\mathrm{I}}, \mathsf{ver}, U)$ - *parse* $sk_{\mathrm{I}} = (pk_{I,\mathsf{DAA}}, sk_{I,\mathsf{DAA}})$. *Receive* $(\mathsf{com}, \Pi_{\mathsf{Join}})$ *from the User. Abort if the proof is invalid, i.e.,* $\mathsf{VerifyProof}(\rho_{\mathsf{Join}}, (\mathsf{com}, pk_{I,\mathsf{DAA}}), \Pi_{\mathsf{Join}}) = 0$. *Otherwise, execute the* $\mathsf{Issue}_{\mathsf{DAA}}$ *protocol with input* $(\mathsf{com}, sk_{I,\mathsf{DAA}})$, *receiving credentials* $cred_U$. *Send* $cred_U$ *to the user.*

$\mathsf{Comment}(pk_{\mathrm{I}}, sk_U, cred_U, dom, m)$ - *set and return* $\gamma = (\sigma, nym, dom, h(m))$ *where* $\sigma \xleftarrow{\$} \mathsf{Sign}_{\mathsf{DAA}}(sk_U, cred_U, dom, h(m))$ *and* $nym \xleftarrow{\$} \mathsf{NymGen}(sk_U, dom) = \mathsf{NymExtract}(\sigma)$.

$\mathsf{Verify}(pk_{\mathrm{I}}, nym, dom, m, \gamma)$ - *Parse* $\gamma = (\sigma, nym, dom, h^*)$ *and* $pk_{\mathrm{I}} = pk_{I,\mathsf{DAA}}$. *Output* 1 *iff* $\mathsf{Verify}_{\mathsf{DAA}}(pk_{I,\mathsf{DAA}}, h^*, dom, \sigma, RL_\emptyset) = 1$, $h(m) = h^*$, $\mathsf{NymExtract}(\sigma) = nym$, *and* $\mathsf{VerifyBsn}(\sigma, dom) = 1$.

$\mathsf{Claim}(pk_{\mathrm{I}}, sk_U, cred_U, dom, m, \gamma)$ - *return evidence* $= \gamma$.

$\mathsf{VerifyClaim}(pk_{\mathrm{I}}, dom, m, \gamma, evidence)$ - *Parse* $\gamma = (\sigma, nym, dom, h)$ *and output 1 iff* $\mathsf{Verify}(pk_{\mathrm{I}}, nym, dom, m, \gamma) = 1$.

The algorithms Setup and KeyGen generate the issuer's DAA keys and parameters for the non-interactive zero-knowledge proof of knowledge for the relation $\mathcal{R}_{\mathsf{Join}}$. The $\mathsf{Join} - \mathsf{Issue}$ protocol closely resembles the $\mathsf{Join} - \mathsf{Issue}_{\mathsf{DAA}}$ protocol of the DAA scheme with two main differences. Firstly, the user provides her secret key as input to the Join algorithm. This is for practical reasons: in Section 4, we explain how this key can be recomputed from a pair of login and password using a key derivation function when a user switches machines. The second difference is the Π_{Join} proof created by the user to ensure honestly generated secret keys and allow the security reduction to extract secret keys generated by the adversary. We remark that during the $\mathsf{Join} - \mathsf{Issue}$ protocol, the user communicates with a publicly known verifier who validates her identity and confirms it to I. In Section 4, we present a protocol for obtaining this confirmation and running a pseudo-probabilistic audit of V by I.

Comment creates the information that U stores on the ledger, consisting of the signed comment γ and pseudonym nym. To provide accountability for censorship, U sends the signature to the ledger, which notifies the website W.

At this point, W must publish the comment $\gamma = (\sigma, nym, dom, m)$ as long as the signature σ, message and *dom* are deemed valid, and *nym* appears exactly once on the ledger.

With the validity requirement on the basename *dom* and the ability to detect repeated basenames in the ledger, we can easily implement the desired throttling mechanism. Let τ be a threshold for some time frame (e.g., a day) and let t mark the current period. Then, a valid *dom* is of the form (t, seq) with $seq \in \{1, \ldots, t\}$. The sequence number *seq* in *dom* is allowed to arrive out-of-order, but it cannot be larger than τ. The throttling is ensured because there exist only τ valid basenames per commenting period and thus only τ valid *nym* per (sk_U, dom).

If W refuses to publish the comment, then U can use Claim to claim censorship and provide the entry on the ledger γ and m as evidence to the public that m ought to be displayed. The public checks the same conditions that W should have applied. Part of this check is to interpret a common agreement for moderation, which we discuss in more detail in Section 4, but do not model explicitly. We show the security of this protocol in the cryptographic model, see the full version [8, Appendix B].

4 Practical implementation

A deployable system needs more than just a cryptographic specification, but a system of incentives and checks. First, we discuss what methods for identity verification are available. We detail how to identity verification can be deferred to the verifiers and misbehaviour can be detected using pseudo-probabilistic audits. A realistic system also has to deal with revocation, which we solve by exploiting a novel property called *updatability*. Finally, we discuss questions related to the end user: how moderation is handled and where to store credentials. Table 1 summarises the protocol components and their security analysis.

Identity providers The verifiers need to attest that only real people receive digital identities and each person obtains only one. We discuss multiple competing solutions to this problem, none perfect by itself. In combination, however, they cover a fair share of the users for our primary target, news websites.

Identity verification services (IVS): Banks, insurers and other online-only services already rely on so-called identity verification services, e.g., to comply with banking or anti-money laundering regulations. Usually, IVS providers verify the authenticity of claims using physical identity documents, authoritative

components	security analysis
base protocol	cryptographic proof ([8, App. B])
encrypted ledger	strictly weakens the attacker
identity verification	formally verified ([8, App. 4])
revocation	simple hybrid argument using [8, App. F]
extended protocol	cryptographic proof ([8, App. C])
storing credentials	trivial modification

Table 1: Overview: security analysis.

name	symbol	purpose	typical duration
epoch	t_e	implicit revocation	one week
billing period	t_b	billing	one month
commenting period	t	throttling	one day

Table 2: Time periods used in protocol.

sources, or by performing ID checks via video chat or post-ID. McKinsey anticipates the market for IVS to reach $16B-20B by 2022 [46]. The business model of these companies revolves around their trustworthiness.

Subscriber lists: Newspaper websites are the main targets of our proposal, because of their political and societal relevance and the moderation cost they are currently facing. It is in their interest to provide easy access to their subscribers. Insofar as bills are being paid, they do have some assurance of the identity of their subscribers, so they can use their existing relationship to bootstrap the system by giving access to their customers right away.

Biometric passports and identification documents: Biometric passports are traditional passports that have an embedded electronic microprocessor chip containing information for authenticating the identity of the passport holder. The chip was introduced to enable additional protection against counterfeiting and identity theft. This authentication process can be performed locally (as part of e.g., border control) or against a remote server. Biometric passports are standardised by the International Civil Aviation Organization (ICAO) [27] and issued by around 150 countries [23].

Encrypting comments on the ledger We distinguish a billing period t_b that is distinct from the commenting period t (see Table 2). Assume a CCA-secure public key encryption scheme ($\mathsf{KG}_{\mathsf{enc}}$, enc, dec), a collision-resistant hash function h and a standard existentially unforgeable digital signature scheme ($\mathsf{KG}_{\mathsf{sig}}, \mathsf{sig}, \mathsf{ver}$). We apply the accountable commenting scheme from Def. 3. The output of $\mathsf{Comment}$ is encrypted with a public key $pk_{\boldsymbol{W},t_b}$ distributed to all websites participating in the current billing period t_b. Claims need to include the randomness used to encrypt. See Fig. 2 for the complete message flow.

Deferring identity verification with pseudo-probabilistic auditing Our security model in the full version [8, Appendix B] abstracts away from the communication between verifier and issuer. We propose a protocol to implement this step and formally verify it in the symbolic setting, which is better suited for reasoning about complex interactions. The protocol (Fig. 3) improves privacy by hiding the identity verification process from the issuer and improves accountability by providing a pseudo-random audit.

We assume a collision-resistant one-way hash function h to instantiate a binding commitment scheme. When a user wants to register, the website directs her to the issuer. They run an authentication protocol akin to the ASW protocol for fair exchange where, in the end, U gets V's signature on a commitment c_I generated by I. Only with this signature, the issuer runs the $\mathsf{Join}-\mathsf{Issue}$ procedure from Def. 3 (repeated in Fig. 4 for completeness). Note that the ledger distributes the issuer's public key and public parameters. In Section 4, we explain a revocation mechanism that is based on updating the issuer's public key every epoch and publishing the fresh key in the ledger. U also makes use of the ledger by storing its credentials in case it needs to recover its state (see Section 4).

After verification, I may trigger a pseudo-random audit by sending the previously hidden values sid, r_I in the commitment c_I of the identity verification

0. U can restore sk_U from *login* and pw,
   and download $h(login), cred_{t_e}$ from L:
   $sk_U := kdf(login, pw)$
   $L \to U : \{h(login), cred_{t_e}\}$
1. U computes the basename from date and sequence:
   $dom := (t, seq)$
2. U computes the nym from sk_U and dom:
   $nym := \mathsf{NymGen}(sk_U, dom))$
3. U signs the hash of his comment:
   $\sigma := \mathsf{Sign}(sk_U, cred, dom, h(m), W)$
4. U encrypts σ, attaches metadata and sends it to L:
   $\gamma := \{enc(pk_{\boldsymbol{W},t_b}, (\sigma, nym); r), h(m), W, dom\}$
   $U \to L : \gamma$
5. L notifies W and U sends the raw comment to W:
   $L \to W : \gamma$
   $U \to W : m$
6. W decrypts γ and verifies the following:
   σ valid, $\mathsf{VerifyBsn}(\sigma, dom) = 1$, $seq \leq \tau$, m acceptable.
7. W queries L with nym:
   $nym := \mathsf{NymExtract}(\sigma)$
   $W \to L : nym$
8. W publishes m if nym fresh and m acceptable.
9. U claims censorship to public, if m not published:
   $claim_U := \{\sigma, nym, r, m\}$
   $U \to public : claim_U$

Fig. 2: Message flow for commenting. In step-0, the user's secret DAA key is restored using a password, see Section 4, and the entries in the ledger are encrypted. Also, we identify the comment m by its hash to save space on the ledger.

1. U creates a secure channel with I, with session id sid:
   $I \to U : sid$
2. U chooses $login$, pw and random r_U, sends to I:
   $r_U \leftarrow\!\!\$ \{0,1\}$
   $U \to I : login, h(r_U, nbd, 1)$
3. I chooses random nonce r_I, creates a commitment c_I, signs it and sends it to U:
   $r_I \leftarrow\!\!\$ \{0,1\}$
   $c_I := h(r_I, sid, h(r_U, nbd, 1))$
   $I \to U : sig(sk_I, c_I)$
4. U creates a secret key from his $login$ and pw:
   $sk_U := kdf(login, pw)$
5. U creates a secure channel with V and sends:
   $U \to V : \{nbd, c_I, r_U, sig(sk_I, c_I)\}$
6. V verifies U's identity with evidence E,
   signs the commitment and sends it to U:
   $\psi := sig(sk_V, c_I)$
   $V \to U : \psi$
7. U recreates the secure session with the previous sid,
   and sends the commitment (signed by V) to I:
   $U \to I : \psi$

** U and I run $\mathsf{Join-Issue}$ protocol (Fig. 4)
** I use ψ to start auditing with V (Fig. 6)

Fig. 3: Identity verification protocol specification

protocol to V (see Fig. 6). If the hash of these values matches the hash of V's signed commitments, an audit is triggered. If we consider a random oracle in place of the hash function, the probability of an audit is $\Pr[\text{audit}] = 2^{-L}$, where L is the number of bits both parties compare. L is agreed upon in advance, to define this probability. Since the nonce r_I has been revealed to V before, I cannot modify the second hash (s') to avoid audit. As the digital signature scheme is existentially unforgeable, I cannot fabricate a valid signature to raise the probability of an audit and to learn something about U. If the session is chosen for audit, V has to hand over the evidence $\{\text{E}\}$ it collected for identification — this is a standard procedure for IVS. If V fails to comply, then I can publish a *claim* and the public can determine whether to audit V.

Presuming that I is honest, the probability that colluding U and V can create n usable fake identities is thus bound by $(1 - \Pr[\text{audit}])^n + \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\lambda)$.

The auditing protocol is very simple cryptographically, but has many possible message interleaving. It is well known that pen-and-paper proofs for such protocols are not only tedious, but also prone to errors. We analyse the protocol in the symbolic model, using the SAPIC process calculus [30] and Tamarin protocol verifier [40]. We formally verify that:

1. U downloads pk_I from L :
   $L \to U : pk_I$
2. U and I run $\mathsf{Join-Issue}_{\mathsf{DAA}}$ proto. using sk_U and pk_I for epoch t_e, and U gains $cred_{t_e}$ and w_{t_e}.
3. U inserts $h(login)$ and $cred_{t_e}$ into L :
   $U \to L : h(login), cred_{t_e}$
4. I inserts $h(login)$ and w_{t_e} into L :
   $I \to L : h(login), w_{t_e}$

Fig. 4: $\mathsf{Join-Issue}$ protocol specification

1. I announces new epoch t'_e and updates her pk_{I,t'_e} :
   $I \to L : t'_e$
2. I asks all Vs, to report all valid *login*s to be updated :
   $V \to I : \mathsf{sig}(sk_V, (\texttt{'update'}, c_I))$
3. I obtains *update message* u for all valid *login*s from L :
   $L \to I : h(login, u)$
4. I creates new credentials for each *login* :
   $cred_{t'_e} := \mathsf{Issuer}_{\mathsf{DAA}}(sk_I, pk_I, u)$
5. I stores new credentials for each *login* in L :
   $I \to L : h(login, cred_{t'_e}, t'_e)$

Fig. 5: Certificate update protocol specification

1. I sends sid and r_I to V:
   $I \to V : \{sid, r_I\}$
2. I and V both calculate two hashes:
   $s := h(r_i, sid, 2)$
   $s' := h(\psi)$
3. Both compare the first L bits of these two hashes:
   $s|_L^0 = s'|_L^0$
4. If the session is chosen for audit, V sends E to I:
   $V \to I : E$
5. If V fails to comply, I can publish a *claim*:
   $claim_I := \{r_I, sid, h(r_U, nbd, 1), \psi\}$
   $I \to public : claim_I$
6. The public audits V, V proves it acted in good faith:
   $V \to public : \{r_U, nbd, E\}$
7. The public gives the verdict.

Fig. 6: Auditing protocol specification

1. Whenever I accepts to run the $\mathsf{Join-Issue}$ protocol with a user, V has validated her identity, unless I or V are dishonest.
2. When determining the need for an audit, neither a dishonest I, nor a dishonest V can predict the value of the other party, unless both are dishonest. Therefore, they cannot trigger or avoid the audit.
3. If the public accepts a claim, then V did indeed receive the values r_I and sid and send out ψ (unless V is dishonest and tricks itself into the obligation of an audit). As these values determine both hashes, the public can now decide if an audit was justified.

The verification takes about 10 sec on a 3.1 GHz Intel Core i7 and 16 GB RAM computer. [7]

Revocation In case U runs the identification protocol a second time with a different V, or simply forgets her password and needs to re-identify, her previous DAA key $sk_{U,\mathsf{DAA}}$ needs to be revoked. But how can U revoke her DAA key if she forgets her password? We circumvent this problem by *implicit revocation*: DAA keys are short-lived by default, but the system can issue new keys without interacting with the user. Keys that are not issued are thus implicitly revoked by the end of their lifetime, which we call the *epoch* (see Fig. 2).

At the start of each epoch t_e, I defines a new public key pk_{I,t'_e} which is chosen so that I can recompute all credentials $cred_{t'_e}$ for the new epoch by itself (see Fig. 5). At this point, only those DAA keys remain valid, for which such a *cred* is computed, all others are implicitly revoked. If a user forgets her password, she

[7] See the full version [8, Appendix E] for the model code.

reports to the verifier, who confirms (by means of the commitment c_I) that her old key is invalidated. Starting from the next epoch, she can use her new key. To allow for such mechanism, the DAA scheme has to be structured in a way that I can update her public key and all users 'credentials without any interaction.

Brickell and Li's scheme with a minor modification possess these features (see the full version [8, Appendix F] for a formal proof).

Updatability is interesting on its own: it allows for regular, non-interactive key rollovers in DAA. I can create each user's credential offline, so the user can fetch this credential (in encrypted form) at later point, even if I is offline.

Holding the issuer accountable In TrollThrottle, a corrupt issuer and verifier can collude to introduce arbitrarily many valid credentials into the system. This form of Sybil attack is difficult to counter while retaining the user's privacy: Without trust in either the verifiers or the issuer, the only way of determining whether a user is legitimate is to have another entity (e.g., the websites, or the public) check this identity — otherwise, the adversary controls all parties involved. Even if done in a pseudo-random manner similar to the auditing procedure in Section 4, the loss of individual privacy would be considerable.

In the full version [8, Appendix C] we present the extended TrollThrottle protocol to mitigate this issue to the extent possible. Here, for every user that joins, a *genesis block* is added to the ledger. This block is signed by the verifier, which allows the public to tell how many credentials were validated by each verifier. Large-scale fraud could thus be detected through an unusual number of participants coming from a single verifier. This information is public and can be computed by any participant at any time.

During the commenting phase, U downloads a subset of genesis tuples[8] and computes a zero-knowledge proof that her genesis tuple is part of this set. She includes this proof along with the time point at which she queried the list in her DAA signature. In the full version [8, Appendix C.5], we show that for Brickell and Li's scheme [14], we can instantiate a non-interactive proof of knowledge system with proofs that are logarithmic in the number of genesis tuples in the ledger. We show that, in addition to the security properties in the full version [8, Appendix B], no adversary can create comments that cannot be attributed.

Other considerations News websites need to moderate comments (see step 8 in Fig. 2). This decision is ultimately a human decision, but it should be based on a binding agreement between the websites and applying laws.

Also, many users expect a system where they can log in from any platform. We, therefore, allow users to restore their identities, by making the users' secret keys e.g., sk_U derivable from their login and password chosen by themselves in the identification process. Hence, we assume there exists an efficient key-derivation function *kdf* that maps to the space of secret keys. Such a function exists for the scheme we use, where the secret key is just an element in $\mathbb{Z}_q^*$.

[8] To achieve, e.g., anonymity among 100 users, about 49 KB of data is downloaded once per commenting period.

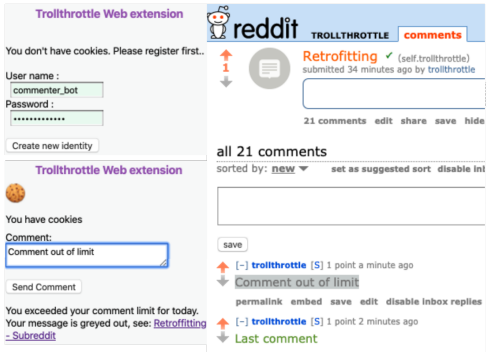

Fig. 7: Screenshot of Reddit deployment, for identity creation and commenting scenarios, see Retrofitting subreddit

measure		mean	median	variance
issuing (on U)[1]	$\delta_U^{\mathsf{Issue}}$	0.038	0.036	0.069
issuing (on W)[1]	$\delta_I^{\mathsf{Issue}}$	0.010	0.009	0.0006
commenting[2]	$\delta^{\mathsf{Comment}}$	0.036	0.032	0.0003
verification	δ^{Verify}	0.021	0.018	0.0002
latency[3]	$t_f - t$	0.022	0.019	0.0002
commenting (on U)[4]	$\delta_U^{\mathsf{Comment}}$	0.058	0.057	0.01

(1) over all new users.(2) computation overhead w/ pre-computed signatures.(3) shows server-side total processing time.(4) on 1000 samples, single-threaded.

Table 3: Evaluation for Reddit use case (3 cores).

The secret key sk_U can be recomputed with the *kdf* and the DAA credentials *cred* can be recovered from the ledger by querying with the hash of the login. Note that the login should not identify the user on other platforms, otherwise an attacker can use it to check if the user is participating in TrollThrottle. The last value of *seq* can be recovered by using bisection to discover the largest *seq* s.t. $\mathsf{NymGen}(sk_U, (t, seq))$ is on the ledger.

5 Evaluation

We evaluate TrollThrottle in terms of how easy it is to deploy, and how much performance overhead it incurs. To demonstrate the former, we retrofit it to an existing website, without any modification to the server-side code — in fact, without the website being aware of this. To demonstrate that it incurs only modest costs, we simulate realistic traffic patterns using a recorded message stream and measure computational overhead and latency.

Deployability We demonstrate that the protocol can be deployed easily by retrofitting it, without any server-side changes, to Reddit.com, the most visited news website in the world [7] and an alleged target for large-scale astroturfing and propaganda efforts [43].

On Reddit, we created a forum as a testing ground. We implemented signature creation and verification in a JavaScript library and used a simple browser extension [5] to load this library when entering the forum. In an actual deployment, this library would be loaded via JavaScript inclusions. We point out, however, the known problem that there is no guarantee the website W is transmitting the correct script. This is a well-known issue for all web-based services that claim end-to-end security and sometimes mitigated by offering optional plugins (e.g., mega.co.nz). We also present the cryptographic implementation details of the simulation [6] in the full version.

Any comment posted in this subreddit is transmitted according to the protocol (see Fig. 2). As the server side is not validating the comments in this instance, this task is performed by the JS library as well. It communicates with a simple HTTP server implementing the public ledger. Comments that do not pass are greyed out by using a subreddit-specific stylesheet (see Fig. 7).

Performance To evaluate TrollThrottle's performance, we compiled three realistic datasets [2] to represent plausible scenarios. Our focus is on traditional news outlets that want to establish a close relation with their readership. We thus examine two scenarios in this domain, and a third, representing an extreme case: the entirety of Reddit, the largest website categorised as 'News' by Alexa [7]. We use Reddit to retrieve realistic commenting patterns for the following scenarios (more details in the full version [8]): (1) Scenario I: nationwide news source News websites operating on national scale have sharp traffic patterns, e.g., the users in the same time zone. We take Germany as an example and simulate the traffic patterns of the German-speaking r/de subreddit with a volume of 168k comments. (2) Scenario II: international newspaper We collect all comments on submitted links to nytimes.com over two months to reach 268k comments and aggregate them to a 24h period. (3) Scenario III: Number of comments per day on Reddit From a 10-year dataset that includes all comments ever posted on Reddit, we pick the recent busiest day, which is 27 June 2019 with 4 913 934 comments.

Performance measures We focus on the performance requirements from the perspective of the news outlet that has to serve users within a given latency and compute the additional cost due to the new computations. To get a precise measure of the overhead incurred, our experiment only simulates the cryptographic operations and does not display the comments or use network communication. The computation is performed separately for the server and the client. We assume the issuer is trusted and thus disregard the extension in the full version [8, Appendix C].

As for the other datasets, we collected the comments annotated with their author's nickname and the time point they were posted. The dataset is thus a sequence of tuples (t, u, m) ordered by the time point t at which u posted comment m. We assume each nickname corresponds to a different actual person, thus over-approximating the effort for key generation. For each (t, u, m), we (1) simulate the issuing protocol, if u comes out in the entire (10 years) dataset for the first time, (2) simulate the commenting protocol to produce a signature for the comment, and finally (3) simulate the server side signature verification.

Step (1) and (2) can be done in a pre-processing step, as they are computed by the user and issuer. We measure the time for commenting ($\delta^{\mathsf{Comment}}$) and issuing ($\delta_I^{\mathsf{Issue}}$ and $\delta_U^{\mathsf{Issue}}$, for the issuer and the user, respectively). For step (3), we simulate the load of the server side on a Ruby-on-Rails application with Nginx load balancer.

For each point (t, u, m) in the database, we simulate the arrival of the encrypted signature (γ, nym) resulting from pre-processing m, at time $t+\delta^{\mathsf{Comment}}+$

scenario	#comments	#cores	daily cost	max. latency	latency < 0.1s	#genesis tuples	ledger size(MB)
Nationwide newspaper (r/de)	168k	1	$1.20	0.166s	99,99%	13,975	204
International news. (url:nytimes)	268k	1	$1.20	0.391s	99.99%	87,223	633
Reddit (r/all)	4.9M	3	$3.60	1.011s	99.99%	1,217,761	10628

Table 4: Scenarios for performance evaluation, including the number of comments, source of the data stream, number of Intel E5 2.6 GHz cores, operating cost per day, maximum latency, percentage of queries answered within 0.1 secs, number of genesis tuples computed (i.e., number of distinct nicknames), and total ledger size.

$\delta_I^{\mathsf{Issue}} + \delta_U^{\mathsf{Issue}}$. We run Verify on the signature and measure the finishing time t_f, as well as the actual processing time δ^{Verify}. We report the results in seconds for the largest dataset in Table 3.

In Table 4, we report the number of cores needed and the cost incurred by the computations just described, i.e., the overhead compared to normal website operations. The number of cores to meet the latency requirement was estimated as described above and used in the simulation. To account for the cost, we employ the core hours metric, which is the product of the number of cores and the total running time on the server. We take Amazon on Demand EC2 pricing [1] as an example and assume $0.05 per core hour. We also report on the maximal latency encountered in the simulation and the percentage of comments that met the target latency of $\leq 0.1s$. Finally, we report the number of genesis tuples created in the ledger, i.e., the number of nicknames in the dataset, and the total size of the ledger, representing an over-approximation of the storage requirements of a single day of operation.

Since comments are hashed before signing, the communication overhead is approximately 2.4 KB, independent of the comment size. To evaluate the storage requirements on a consensus-based public ledger, we chose Tendermint [24] as an example. Tendermint employs a modified AVL tree to store key-value pairs. Values are kept in leaf nodes and keys in non-leaf nodes. The overhead is about 100 bytes per non-leaf node [4]. For the largest dataset, each participant in Tendermint would thus require approximately 12 GB of space. Once the current commenting period is over, the signed comments and hence most of the data can be purged. To allow accountability for censorship over the last month, the data of the last thirty commenting periods can be stored on less than 0.5 TB.

In summary, the additional cost on the websites is modest compared to the moderation effort saved.

6 Limitations

Despite the auditing by the issuer and the limited accountability for colluding issuer and verifiers in the extended protocol, we have centralised trusted au-

thorities. One way to remove these is to introduce protocols that can recognise Sybils. This could relieve the issuer from the responsibility of auditing the verifiers and potentially allow for a protocol with accountability features to deter misbehaviour. As this topic is orthogonal to our protocol, we leave it for future work, but remark that, theoretically, Sybil-detection is possible without user identification. A potential approach is to combine biometric methods [9,41] with captchas. Uzun and Chung proposed such a protocol to show liveness. Here, the user's response to a captcha involves physical actions (smiling, blinking) that she captures in a selfie video [45] within a 5s time limit. Their approach is based on the fact that automated captcha-solving takes considerable time, and face reenactment (e.g., [42]) is difficult to do at scale. Building on the same assumptions, a Sybil-detection scheme could be built by pseudo-randomly defining sets of users that need to show liveness *at the same time*.

TrollThrottle aims to provide a similar user experience to website logins. Hence, all client-side secrets are derived from the login and password of the user and thus vulnerable to password-guessing attacks. This can be mitigated by incorporating a two-factor authentication into the protocol, or by setting up the key generation to require a password of sufficient length and entropy, as to enforce the use of password managers.

Finally, the client-side code is loaded by the website, which could potentially include a different script albeit this behaviour would leave traces. As previously discussed (see Section 5), this is a well-known problem for web-based apps, and usually mitigated by offering optional plugins.

7 Related work

The detection of astroturfing has been tackled using reputation systems (e.g., [37]), crowdsourcing (e.g., [47]) and text analysis (e.g., [38]). Fundamentally, the posting profile of a politically motivated high-effort user is not very different from a state-sponsored propagandist [29], hence we focus on prevention instead of detection. The detection and prevention approaches could be combined, but detection approaches either come at a loss of accountability, or they need to explain their decisions, although many of them rely on the fact that the bot is not adapting to the mechanism (e.g., via adversarial machine learning).

Our approach is similar to anonymity protocols in which we specify a way of exchanging messages without revealing identities. In contrast to anonymity protocols, TrollThrottle provides anonymity with respect to the ledger, but presumes the communication channels to provide sufficient anonymity. By itself, TrollThrottle is not resistant against traffic analysis — here anonymity protocols come into play. One might ask whether anonymity protocols already do what TrollThrottle proposes to do. To the best of our knowledge, Dissent [19] is the only anonymity protocol that provides explicit accountability guarantees, but these pertain to the type of communication, not to sending more messages than allowed. Furthermore, unlinkability is not achieved within the group, but towards outsiders.

Pseudonymity systems like Nym [25] or Nymble [28] provide anonymous, yet authenticated access to services, but some allow resource owners to block access at their discretion. By using a ledger and a common set of rules, TrollThrottle users can claim and prove censorship, but have to trust the ledger. This is in contrast to p2p-protocols, where censors may be sidestepped, but cannot be forced to publish the content themselves. Dingledine et al. advocate for the transaction of reputation/credit between pseudonyms [20]. By contrast, the credit in our scheme is essentially the number of nyms. This simplifies the system and ensures unlinkability, at the cost of inherent limitations: the 'credit' is the same for every participant (τ for each commenting period) and cannot be transferred.

One of the main cryptographic components of TrollThrottle is a specific DAA scheme with additional properties (instant-linkability and updatability). DAA was introduced as a way to address privacy issues of the remote attestation protocol proposed for TPMs. We focused on the scheme by Brickell and Li [14], because it supports these properties, produces short signatures and because a reference implementation was available. Other DAA schemes(e.g., [12,13]) may also provide these properties.

There are building blocks besides DAA that are compatible with TrollThrottle. Anonymous Credentials (AC) allow users to prove (a set of) attributes about themselves to third parties, usually via an interactive protocol (but there are non-interactive schemes). Single-show schemes (e.g., [10]), would require a fresh credential for each comment the user would like to post in the future. Multi-show schemes (e.g., [15]) mitigate this issue, but a user would still need a unique *attribute* per day and sequence number – this would allow the issuer to link comments. Therefore, lightweight AC schemes are not suitable – a fitting AC scheme needs to support domain-specific pseudonyms with a secret-key based attribute. Indeed, DAA can be viewed such a credential system with the DAA key as the attribute.

The most similar credential system to the DAA scheme, that we used, was proposed by Camenisch et al. [16]. In this system, an issuer creates and distributes so-called dispensers. Dispensers are used to create a pre-defined number of one-time credentials valid for a given date. This system can be immediately used in TrollThrottle. As an implementation was not available, we perform a qualitative analysis. On the one hand, verification is faster in their scheme, they perform seven multi-exponentiations in a prime order group and one in an RSA group, while Brickell and Li's scheme perform one multi-exponentiation in each group i.e., $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and one pairing computation. On the other hand, the signatures, which consists of a unique serial number (similar to a pseudonym) and a number of proofs of consistency are at least twice as much larger and their size depend on how the proofs are implemented. This produces considerable computation and communication overhead in the ledger. Moreover, the verification of comments is performed by the websites, making verification efficiency less important than the size of the data included in the ledger. Therefore, the DAA scheme represents a preferable tradeoff.

8 Conclusion

The prevalence of social bots and other forms of astroturfing in the web poses a danger to the political discourse. As many newspapers are closing down their commenting functionality despite the availability of sophisticated detection methods, we argue that they should be combined with a more preventive approach.

We presented TrollThrottle, a protocol that raises the cost of astroturfing by limiting the influence of users that emit a large amount of communication, even if using different pseudonyms. TrollThrottle preserves anonymity, provides accountability against censorship, it is easy to deploy and comes at a modest cost. We also discuss its social impact in the full version [8, Appendix D].

By how much do we raise the cost of astroturfing? We shall regard the last week before the 2016 US election for a rough calculation. The computational propaganda project considered around 3.4M election-related tweets to be originating from bots who emit more than 50 messages per day [26]. If we assume a threshold of 20 messages/day and perfect coordination between the bots, 24 178 identities need to be stolen to reach the same target. A lab study [11] finds that users are willing to sell their Facebook accounts for $26 on average, which is only slightly above the black-market value for stolen verified Facebook accounts. Such operation would thus face a cost of $634 501 and a risk of detection.

Acknowledgements: This work has been partially funded by the German Research Foundation (DFG) via the collaborative research center "Methods and Tools for Understanding and Controlling Privacy" (SFB 1223), and via the ERC Synergy Grant IMPACT (Grant agreement 610150). The first author holds the Turkish Ministry of National Education Scholarship. The second author is supported by the German Federal Ministry of Education and Research (BMBF) through funding for the CISPA-Stanford Center for Cybersecurity (FKZ: 16KIS0762).

References

1. Amazon EC2 on-demand pricing, https://aws.amazon.com/ec2/pricing/on-demand/
2. Big query reddit dataset, https://bigquery.cloud.google.com/dataset/fh-bigquery:reddit_comments
3. Department of justice : Charges in case 1:18-cr-00032-dlf, https://www.justice.gov/file/1035477/download
4. Tendermint avl performance and benchmarks, https://github.com/tendermint/iavl/blob/master/PERFORMANCE.md
5. Trollthrottle Browser Extension, https://github.com/iesiyok/trollthrottle_chrome
6. Trollthrottle Simulation, https://github.com/iesiyok/trollthrottle
7. Alexa News (2019), https://www.alexa.com/topsites/category/Top/News
8. Anonymized.: Trollthrottle — raising the cost of astroturfing (2019), http://supplement.bplaced.net/trollthrottle-full.pdf
9. Azimpourkivi, M., Topkara, U., Carbunar, B.: A secure mobile authentication alternative to biometrics. In: Proc. of the 33rd ACSAC. pp. 28–41. ACSAC 2017, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3134600.3134619, http://doi.acm.org/10.1145/3134600.3134619

10. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: (CCS'13). pp. 1087–1098. ACM (2013)
11. Benndorf, V., Normann, H.T.: The willingness to sell personal data. The Scandinavian Journal of Economics **120**(4), 1260–1278 (2018)
12. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: CCS'04. pp. 132–145. ACM (2004)
13. Brickell, E., Chen, L., Li, J.: A new direct anonymous attestation scheme from bilinear maps. In: Trusted Computing-Challenges and Applications, pp. 166–178. Springer (2008)
14. Brickell, E., Li, J.: A pairing-based daa scheme further reducing tpm resources. In: Acquisti, A., Smith, S.W., Sadeghi, A.R. (eds.) Trust and Trustworthy Computing. pp. 181–195. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
15. Camenisch, J., Drijvers, M., Dzurenda, P., Hajny, J.: Fast keyed-verification anonymous credentials on standard smart cards. In: ICT Systems Security and Privacy Protection (SEC 2019). pp. 286–298. Springer (2019)
16. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: CCS'06. pp. 201–210. ACM (2006)
17. Carroll, O.: St.Petersburg troll farm to influence US election campaign, https://www.independent.co.uk/news/world/europe/russia-us-election-donald-trump-st-petersburg-troll-farm-hillary-clinton-a8005276.html
18. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OSDI. vol. 99, pp. 173–186 (1999)
19. Corrigan-Gibbs, H., Ford, B.: Dissent: accountable anonymous group messaging. In: CCS'10. pp. 340–350. ACM (2010)
20. Dingledine, R., Mathewson, N., Syverson, P.: Reputation in p2p anonymity systems. In: Workshop on economics of peer-to-peer systems. vol. 92 (2003)
21. Etim, B.: Why no comments? it's a matter of resources, https://www.nytimes.com/2017/09/27/reader-center/comments-moderation.html
22. Ferrara, E., Varol, O., Davis, C., Menczer, F., Flammini, A.: The rise of social bots. Commun. ACM **59**(7), 96–104 (Jun 2016). https://doi.org/10.1145/2818717, http://doi.acm.org/10.1145/2818717
23. Gemalto: The electronic passport in 2018 and beyond (Jun 2018), https://www.gemalto.com/govt/travel/electronic-passport-trends
24. Herlihy, M., Moir, M.: Enhancing accountability and trust in distributed ledgers (06 2016)
25. Holt, J.E., Seamons, K.E.: Nym: Practical pseudonymity for anonymous networks. Internet Security Research Lab Technical Report **4**, 1–12 (2006)
26. Howard, P., Kollanyi, B., Woolley, S.C.: Bots and automation over twitter during the second us presidential debate. Tech. rep., Political Bots (2016)
27. ICAO: Machine Readable Travel Documents - Part 11: Security Mechanism for MRTDs. Doc 9303 (2015)
28. Johnson, P.C., Kapadia, A., Tsang, P.P., Smith, S.W.: Nymble: Anonymous ip-address blocking. In: Intern. Workshop on PETS. pp. 113–133. Springer (2007)
29. Kreil, M.: Social bots, fake news und filterblasen (2017), https://en.wikipedia.org/wiki/List_of_newspapers_by_circulation
30. Kremer, S., Künnemann, R.: Automated analysis of security protocols with global state. In: S&P'14. pp. 163–178. IEEE Comp. Soc. (2014). https://doi.org/10.1109/SP.2014.18

31. Kumar, S., Cheng, J., Leskovec, J., Subrahmanian, V.: An army of me: Sockpuppets in online discussion communities. In: Proc. of the 26th Int. Conf. on WWW. pp. 857–866 (2017)
32. Kuran, T., Sunstein, C.R.: Availability cascades and risk regulation. Stan. L. Rev. **51**, 683 (1998)
33. Leiser, M.: Astroturfing, cyberturfing and other online persuasion campaigns. EJLT **7**(1) (2016), http://ejlt.org/article/view/501
34. Li, Y., Ye, J.: Learning adversarial networks for semi-supervised text classification via policy gradient. In: Proc. of the 24th ACM SIGKDD. pp. 1715–1723. KDD '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3219819.3219956
35. Long, K.: Keeping the times civil, 16 million comments and counting, https://www.nytimes.com/2017/07/01/insider/times-comments.html
36. Michael Bristow, BBC News, B.: China's internet 'spin doctors' (2008), http://news.bbc.co.uk/2/hi/7783640.stm
37. Ortega, F.J., Troyano, J.A., Cruz, F.L., Vallejo, C.G., Enríquez, F.: Propagation of trust and distrust for the detection of trolls in a social network. Computer Networks **56**(12), 2884 – 2895 (2012). https://doi.org/https://doi.org/10.1016/j.comnet.2012.05.002, http : / / www . sciencedirect.com/science/article/pii/S138912861200179X
38. Peng, J., Choo, R.K., Ashman, H.: Astroturfing detection in social media: Using binary n-gram analysis for authorship attribution. In: 2016 IEEE Trustcom/BigDataSE/ISPA. pp. 121–128 (Aug 2016). https://doi.org/10.1109/TrustCom.2016.0054
39. Reuter, M., Dachwitz, I.: Moderation bleibt handarbeit: Wie große online-medien leserkommentare moderieren, https://netzpolitik.org/2016/moderation-bleibt-handarbeit-wie-tageszeitungen-leserkommentare-moderieren
40. Schmidt, B., Meier, S., Cremers, C., Basin, D.: The tamarin prover for the symbolic analysis of security protocols. In: CAV'13. LNCS, vol. 8044, pp. 696–701. Springer (2013)
41. Sluganovic, I., Roeschlin, M., Rasmussen, K.B., Martinovic, I.: Using reflexive eye movements for fast challenge-response authentication. In: Proc. of the 2016 ACM SIGSAC CCS. pp. 1056–1067. ACM (2016)
42. Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., Nießner, M.: Face2face: Real-time face capture and reenactment of rgb videos. In: Proc. of the IEEE CVPR. pp. 2387–2395 (2016)
43. Tony Romm, W.P.: Senate investigators want answers from reddit and tumblr on russia meddling (2018)
44. Twitter, I.: Update on twitter's review of the 2016 US election, https://blog.twitter.com/official/en_us/topics/company/2018/2016-election-update.html
45. Uzun, E., Chung, S.P.H., Essa, I., Lee, W.: rtcaptcha: A real-time captcha based liveness detection system. In: NDSS. Georgia Institute of Technology (2018)
46. Vikram Iyer, M..C.: The next $20 billion digital market — id verification as a service (2018), https://fuelbymckinsey.com/article/the-next-20-billion-digital-market-id-verification-as-a-service
47. Wang, G., Mohanlal, M., Wilson, C., Wang, X., Metzger, M., Zheng, H., Zhao, B.Y.: Social turing tests: Crowdsourcing sybil detection. arXiv preprint arXiv:1205.3856 (2012)
48. Wullner, D.: Lassen sie uns diskutieren, https://www.sueddeutsche.de/kolumne/ihre-sz-lassen-sie-uns-diskutieren-1.2095271