CROCUS: CROwd Counting Using Smartphones

Daniel Bosk[1], Simon Bouget[1], Sonja Buchegger[1], and Sébastien Gambs[2]

[1]KTH Royal Institute of Technology,
{dbosk,bouget,buc}@kth.se
[2]Université du Quebec à Montréal, sebastien.gambs@uqam.ca

April 29, 2020

Abstract

Crowd counts vary according to who is doing the counting and what methods and criteria they use. Current methods have wide margins for error, are difficult to verify, and can be privacy-invasive. We recognize the potential adversarial setting for crowd counts and thus aim for both transparency (in the form of verifiability) and privacy. In this paper we propose CROCUS, a decentralized system based on smartphones that combines anonymous credentials, witnesses of proximity, and storage of participation proofs on a public ledger (e.g., blockchain) to achieve properties similar to those needed for electronic voting, where, put simply, it is important to count everyone, but only once, not leak private information, and be able to prove that the count was done correctly. We find that, with some assumptions about the availability of future technology such as distance-bounding chips on smartphones, CROCUS can provide the desired properties at acceptable performance.

Contents

1 Introduction

While our proposal is meant for crowd counting in general and to be applicable to any kind of event, we represent an event with a political protest against an oppressive regime throughout the paper. We choose this instantiation of an event because it is the most challenging in terms of requirements for privacy and verifiability (transparency).

Historically, there are many examples of protests in which the count estimated by police and that of the organizers differ significantly, sometimes by hundreds of thousands. Even without foul play, the difference is quite natural as both parties have different objectives and metrics. More precisely, the organizers want to count everyone who participated while the police want to estimate the count at the peak of participation, due to crowd control [44]. Among the numerous recent examples in which it is difficult to establish the actual number of participants, there are the demonstrations against the president in South Korea [44], Trump's inauguration [38], the 2017 Women's March in the US [47], the demonstrations against the change of constitution in Venezuela [30] or for the independence of Catalonia [23].

Consider the following scenario. Alice is an activist that organizes a protest in some location(s) against the current government, represented by Grace. Alice wants to estimate the number of participants to prove a certain support for her cause against Grace in a way that is both demonstrably accurate and does not increase the participants' risk of retribution against them from Grace's regime. To realize this objective, a reliable yet privacy-preserving crowd-counting mechanism is needed, which to the best of our knowledge is a problem that has not yet been entirely solved.

Existing methods for crowd-counting vary significantly in terms of approaches. Most of them, however, lack precision (i.e., they have large error margins) and can only give an estimate for a particular snapshot in time, not the cumulative participation count — at least not without counting some persons multiple times. In addition, they lack verifiability in the sense that one has to trust the third party responsible for implementing the counting method.

Finally, one important observation about crowd-counting that has not been adequately addressed in the design of current crowd-counting solutions is that it actually is an adversarial setting. Alice the activist has an incentive to increase the tallied number of participants, whereas Grace (and possibly other entities) has an incentive to decrease it. In this paper, our main objective is to provide a scheme preventing both Alice and Grace from cheating by providing verifiable participation counts that can resist Sybil attacks while still preserving the participants' privacy to the extent possible given their physical presence at the protest. While one cannot prevent an observer (physically present or looking at photos or videos) from recognizing a particular individual at a protest, we do not want the digital traces of our protocol to increase any risk for the participants.

The paper is organized as follows. In Section 4, we describe our system model and summarize the desired properties for crowd counting, followed by a

discussion of current crowd-counting methods in Section 2. In Section 3, we formalize the notion of protest and the desired verifiability and privacy properties, and in Section 5, we give the relevant background on the building blocks of our solution. We present CROCUS, a privacy-preserving crowd counting protocol in in **??**, analyze its security in Section 8, and estimate its performance in Section 9. We compare it to related work in **??**. Finally, we discuss limitations and assumptions in Section 11 and give our conclusions in Section 12.

2 Desired properties and current crowd-counting methods

2.1 Desired properties

We note that, in general terms, protests are petitions with a given time and location. Protests, petitions and elections share that in all three many individuals express their opinion. These opinions can be sensitive (e.g., be a cause for discrimination or persecution). For that reason we have strong requirements for verification and privacy for elections, it follows that we should have similar properties for protests and petitions.

We draw inspiration from properties for voting systems as formalized in [17]: Daniel ►*Find a new ref for voting verifiability properties, Douglas commented that the last one is not the origin.*◄

Eligibility: anyone can verify that each cast vote is legitimate.

Universal verifiability: anyone can verify that the result is according to the cast votes. Daniel ►*Douglas: technically this is Sako-Kilian, non-technically, it's old.*◄

Individual verifiability: each voter can verify that their vote is included in the result. Daniel ►*Douglas: this is much earlier than Sako-Kilian, in essence Fiat-Shamir (technically). Non-technically, much older.*◄

In our context, votes are translated into *participation proofs*. Universal and individual verifiability remain the same, in the sense that anyone can verify the participation count by counting the proofs and a participant can verify their proof is included. The eligibility requirement is slightly different as for protests it must also include temporal and spatial eligibility (i.e., each participation proof satisfies some temporal and spatial relation to the protest). In essence, the proof must bind the person to the time and location of the protest. (This is the difference to a petition.)

In [17], the main three privacy properties for voting protocols are given as:

Vote privacy: the voting does not reveal any individual vote.

Receipt freeness: the voting system does not provide any data that can be used as a proof of how the voter voted.

Coercion resistance: a voter cannot cooperate with a coercer to prove their vote was cast in any particular way.

Coercion resistance in voting typically relies on physical isolation (e.g., private voting booths), including for digital systems, and that is by definition not possible for public events. For instance, someone could simply physically bring Alice to a protest against her will. As for receipt freeness, while desirable *in itself*, it implies a conflict with verifiability in our context: in contrast to voting, receipt freeness for *how* the voter voted (i.e., the cause of the protest) here implies receipt freeness for *that* the voter voted (i.e., the protester was there), which would make verifiability impossible.

Therefore in our context, the crucial property is vote privacy. More precisely, for the protester we want unlinkability (from the adversary's perspective) between a protester's real identity P and the participation proof (and thus also the protest itself). Phrased differently, given a participation proof, Grace should not be able to distinguish if it was Alice or Bob who participated. Furthermore, if Grace has managed to link one proof to Alice due to some auxiliary knowledge, she should not be able to link it to another proof (from a different protest).

2.2 Current crowd-counting methods

The seemingly most common method for counting crowds at protests is *Jacobs's method* [2, 3, 28, 38, 44]. This manual method devised in the 1960s relies on aerial pictures of the event. The verifier divides the protest venue into regions and then estimates the density of the crowd in the different regions before summing them up to get an estimate of the global count. Using pictures makes it difficult to get cumulative counts, verify that the pictures have not been manipulated, and to have both privacy and individual verifiability: either one is included in the picture (privacy problem) or not (verifiability problem). Similar limitations exist for estimating the number of persons in a picture or video (e.g., the work of [49] or CrowdSize [13]).

Another problem for all the above methods is exemplified by the demonstrations in Seoul: "[t]he demonstrators not only gather in open space, but also small alleys and between buildings" [44]. In this situation it is very difficult to faithfully capture the situation. Taking pictures from different angles risks double counts. Another challenges is determining whether people near the event's perimeter are participants or simply bystanders [37].

Counting MAC addresses, as done by a company during the protests in Seoul [44] suffers from MAC randomization, though some tracking of smartphones could still be possible with a different method [45] or using *IMSI catchers*; none of which is verifiable.

An approach that relies on a trusted infrastructure was recently deployed by a collection of media outlets to count protesters passing the line defined by a trusted sensor on marches [10]. This solution does not offer strong verifiability guarantees and thus is complemented by micro-counts made by humans to estimate their margin of error.

CrowdCount CrowdCount [12] is a web service that lets Alice create an event such that anyone can submit their location to register that they are in Alice's event. Another related approach based on devices is UrbanCount [15], which relies on epidemic spreading of crowd-size estimates by device-to-device communication to count crowds in dense urban environments with high node-mobility and churn. However, there is no consideration of a potentially adversarial setting and thus no verifiability or checks on eligibility. DiVote [14], a prior work by the same authors for polling in dense areas, avoids double counting, but again only works with honest participants and thus does not suit an adversarial setting.

3 Definitions

3.1 Protest, crowd estimation

To be able to estimate the participation count for a protest, we first need to define this concept and which quantity should be counted. Let us start by considering some examples. During the demonstrations against the South Korean president in Seoul in 2016 "[t]he rallies stretch[ed] from midday to late night — some people stay[ed] for several hours, others just several minutes" [44]. These rallies were all in the same location in the capital and repeated every weekend for a few weeks. The Women's Marches in 2017 [47], on the other hand, occurred in parallel in many locations. We also have the Venezuelan demonstrations in 2017 in which "anti-government demonstrators have staged daily protests across Venezuela" [43] while "pro-government workers sang and danced as they staged a rival march to show their support for the president's controversial plan to rewrite the constitution" [30]. Generalizing from these examples, the minimal common part is the cause, while the location (or area) considered varies over time.

For the rest of the paper, we will refer to the organizer as Alice. We assume that the objective of Alice is to count everyone who participated at any time and in any of the locations [44] Formally, we define a protest as an event that is uniquely identified by its cause *cid*, its time interval t and its location (area) l. More specifically, we will use the following definition.

Definition 1 (Subprotest, Protest)**.** *A* subprotest $p = (cid, t, l)$ *is a tuple in which* $cid \in \{0,1\}^\lambda$*, for some fixed* $\lambda \in \mathbb{N}$*, is the identifier of the cause of the protest,* $t \subseteq \mathcal{T}$ *is a time period and* $l \subseteq \mathcal{L}$ *is the location (topological connectedness is not necessary).*

A protest $\mathcal{P}$ *is the set of subprotests sharing the same cid.*

The protests described in the previous examples can be captured using this definition by decomposing them into subprotests. Each subprotest will then be encapsulated by our definition and to estimate the total participation to the protest we can just sum up the estimates obtained. Similarly for marches, the marching path can be divided into subprotests with locations (or areas) that slightly overlap.

Each participant who wants to be counted must submit a *participation proof*. The proof must be associated with the protest (i.e., its cause identifier *cid*), and its time and location must coincide with one of the subprotests.

Our protocol relies on *witnesses* to certify and associate the proof to the time and location by creating a *proof share*. A witness is only allowed to create one proof share per protester to avoid the risk of count inflation. (Note that a participant of a protest can take the role of a protester but also act as witness for other protesters.) Then, the set of all *valid* proof shares forms the participation proof of a protester.

Definition 2 (Valid proof share)**.** *A* proof share $s = (cid, t, l, pid, wid)$ *is a tuple in which:* cid, t, l *are as in Definition 1;* pid *is a protester's pseudonym for the protest identified by* cid*;* wid *is a witness's pseudonym for a protester with pseudonym* pid*.*

Furthermore, we say that s *is* valid *for a subprotest* $p = (cid', t', l')$ *if and only if* $cid = cid', t \subseteq t', l \subseteq l'$ *and denote this by* $s \sqsubseteq p$*.*

We let $\mathcal{S}$ denote the set of all proof shares. Let $\mathcal{S}_\mathcal{P} = \{s \in \mathcal{S} \mid \exists p \in \mathcal{P}, s \sqsubseteq p\}$ be the subset of proof shares related to a protest $\mathcal{P}$. We denote by $=_{pid}$ the equivalence relation on $\mathcal{S}$ where $(cid, t, l, pid, wid) =_{pid} (cid', t', l', pid', wid')$ if $pid = pid'$.

Definition 3 (Participation proof)**.** *We denote by* $\Pi_\mathcal{P} = \mathcal{S}_\mathcal{P} / =_{pid}$ *the* set of all proofs of participation for the protest $\mathcal{P}$*. The* participation proof *of a protester with pseudonym* i *who participates in a protest* $\mathcal{P}$ *is the set*

$$\pi_{i,\mathcal{P}} = \{(cid, t, l, pid, wid) \in \mathcal{S}_\mathcal{P} \mid pid = i\},$$

of all proof shares with the same protester and valid for any subprotest of $\mathcal{P}$*.*

We can now define the participation count as follows.

Definition 4 (Participation count)**.** *We define a* participation count *of a protest* $\mathcal{P}$ *as the cardinality* $|\Pi_\mathcal{P}^{\varsigma,\theta}|$ *of the set of* eligible *participation proofs respectively to a strength function* ς *and a threshold* θ*:*

$$\Pi_\mathcal{P}^{\varsigma,\theta} = \{\pi_{i,\mathcal{P}} \in \Pi \mid \varsigma(\pi_{i,\mathcal{P}}) \geq \theta\}$$

with $\varsigma \colon \mathcal{P}(\mathcal{S}) \to \mathbb{R}_+$ *and* $\theta \in \mathbb{R}_+$*.*

The strength function ς can be used to regulate the trust in the estimated participation count. **Daniel** ▶ *The strength function can take the size of the time interval into account too, smaller time intervals around the event yields stronger proofs.* ◀ In general, ς can be defined as a weighted sum of the proof shares, $\varsigma = \sum \omega_i s_i$, with the weights ω_i being the trust in the witness corresponding to the proof share s_i, and the threshold θ represents the total trust needed to accept a participant as valid. One example would be to set all weights to 1 for ς to return the number of unique witnesses and thus let θ to be the threshold of the number of required witnesses. Another possibility would be to also have

a particular type of witness, called *trusted witness*, participating in the protest. For instance, the role of the trusted witness could be taken by the independent journalist Jane. In this situation, the weights would be 1 for trusted witnesses and 0 for any other witness, and setting $\theta = 1$ would require at least one proof share issued by a trusted witness. Finally, both approaches can be combined by giving a weight of 1 to all non-trusted witnesses and a weight of θ to trusted witnesses. This results in a participant being eligible if they are witnessed either by θ non-trusted witness or by one trusted witness.

Simon ▶*check that the description in section V corresponds*◀

3.2 Verifiability and privacy requirements

We now try to make the properties from Section 2.1 more specific. We define three verifiability requirements, among which eligibility can be further broken up into four subproperties:

V1. *Eligibility*: anyone can verify that each participation proof provides temporal and spatial eligibility and that only one participation proof is counted per individual.

- V1.1. *Temporal eligibility*: demonstrate that the proof was created after the start of the protest and before the end of the protest.
- V1.2. *Spatial eligibility*: demonstrate that the proof is spatially related to the physical location or journey of the protest.
- V1.3. *Counted only once*: A protester can create *one and only one* pseudonym (*pid* in Definition 2) per protest (*cid* in Definition 2), this pseudonym is unique except with negligible probability. Analogously, a witness can create *one and only one* pseudonym per protester (*wid* in Definition 2), this pseudonym is unique except with negligible probability.
- V1.4. *Designated event*: prove that the proof is designated for the particular protest.

V2. *Universal verifiability*: anyone can verify that the result obtained matches the submitted participation proofs.

V3. *Individual verifiability*: each participant can verify that their participation proof is included in the global count.

For privacy, we require a set of unlinkability properties:

P1. *Pseudonym unlinkability*: given a protest (identifier *cid*), protesters Alice and Bob, and a pseudonym pid_b, the adversary cannot tell if $pid_b = pid_{\text{Alice}}$ or $pid_b = pid_{\text{Bob}}$, except with negligible probability. And similarly with *wid* if Alice and Bob act as witnesses.

P2. *Protest unlinkability*: protesters' pseudonyms $(pid_{cid}, pid_{cid'})$ must be unlinkable between protests (cid, cid') from the adversary's perspective.

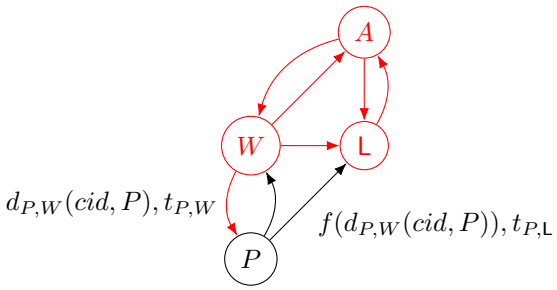

Figure 1: An overview of the adversary model. The protester (with real identity) P and witness (with real identity) W communicate. They exchange protocol data as a function d of the protest and protester, $d_{P,W}(cid, P)$, and record the time it happened, $t_{P,W}$. The protester submits $f(d_{P,W}(cid, P))$, for some function f, to the ledger L, who records the time that happened, $t_{P,\mathsf{L}}$.

P3. *Witness unlinkability*: witnesses' pseudonyms ($wid_{pid}, wid_{pid'}$) must be unlinkable between protesters (pid, pid') from the adversary's perspective.

What these properties say is that pseudonyms must look random (requirement P1) and that each psdeudonym must not be reused more than strictly necessary for the verifiability properties (requirements P2 and P3). In Pfitzmann-Hansen terminology [41], they are role-relationship pseudonyms, but with increasingly narrowed notions of roles and relationships (participant of particular protest, witness of a particular participant at a particular protest). In the terminology of Martucci et al. [36], for each protester a protest is a context or identity domain whereas for each witness every protester is a context (or identity domain).

3.3 Adversary model

There are three players: the protester (with identity) P, a witness (with identity) W, and a ledger L (i.e., time-stamping storage, that will contain the set of all proof shares, $\mathcal{S}$). The protester P and the witness W communicate some protocol data, $d_{P,W}(cid, P)$, and record when the communication occurred $t_{P,W}$. The protester P communicates with L and L only learns some function f of the protocol data exchanged with the witness, $f(d_{P,W}(cid, P))$, and the time of the communication ($t_{P,\mathsf{L}}$). This is illustrated in Fig. 1.

The goal of the adversary is to link a protester's real identity P to a protest identifier cid. The adversary maliciously controls W. The adversary honest-but-curiously controls L, but can submit to L like everyone else (e.g., a malicious protester P). The adversary only learns the protocol data, i.e., what is sent over the channel — no auxiliary data such as who is on the other side of the channel obtained as side information, e.g., by face recognition or inference from address to identity.

4 System model

To set the stage, we will now describe our system model including assumptions and a summary of the desired properties for crowd counting.

4.1 Model and assumptions

Throughout this paper, each time we refer to a participant, such as Alice, we actually mean a personal device that can perform cryptographic operations and communicate with other local devices on Alice's behalf. Furthermore, we assume that each participant has a digital identity certificate signed by a certificate authority (CA)[1] that ensures a one-to-one mapping between an identity and a cryptographic key[2].

In practice, participants witness each other's participation using their smartphones (or similar devices) running the protocol described in **??** and uploading their testimony (i.e., proof shares) to a ledger (such as a blockchain) after the protest. During the protest, the devices might be limited by their batteries and computations they can perform and only have local connectivity to each other. No connection to any global network such as the Internet is necessary at that time. Nonetheless, before and after the protest, we assume that the devices have global connectivity (i.e., Internet connections) and are not computationally limited by any battery, for the participants to be able to upload their proof shares to the ledger.

5 Building blocks

In this section, we will briefly review the primitives forming the building blocks of CROCUS.

5.1 Zero-knowledge proofs of knowledge

We will use the notation introduced by Camenisch and Stadler [9]:

$$\mathrm{PK}\big\{(\alpha,\beta,\gamma) : y = g^{\alpha}h^{\beta} \wedge y' = \hat{g}^{\gamma}\big\}, \tag{1}$$

which means that we prove knowledge of α, β, γ ensuring that y, y' are of the form $y = g^{\alpha}h^{\beta}$ and $y' = \hat{g}^{\gamma}$, respectively. Greek letters are known only to the prover and used for the information for which the prover wishes to prove knowledge, while all other letters are known by the verifier.

When a proof of knowledge is turned into a signature using the Fiat-Shamir heuristic [25], we will denote it as

$$\sigma \leftarrow \mathrm{SPK}\big\{(\alpha,\beta,\gamma) : y = g^{\alpha}h^{\beta} \wedge y' = \hat{g}^{\gamma}\big\}(m),$$

[1] We note that any system that prevents Sybil attacks [22] will do. The CA can be centralized or decentralized, but as Douceur [22] points out, the CA must be *logically centralized* to prevent Sybil attacks.

[2] For every physical person there exists at most one valid digital certificate.

which yields a signature σ on m, ensuring that the issuer knows α, β, γ such that $y = g^{\alpha} h^{\beta}$ and $y' = \hat{g}^{\gamma}$.

5.2 Sybil-free pseudonyms

We must protect against Sybil attacks [22]. We use the idea of identity domains and self-certified Sybil-free pseudonyms proposed by Martucci et al. [35]. However, we don't need the ability to (nor want to be able to) deanonymize pseudonyms, so we don't use the entire e-token dispenser scheme by Camenisch et al. [7] that Martucci et al. [35] use, but rather just what they call the token serial number. More specifically, what we need is an anonymous credential system with the following properties.

We need an anonymous credential system, AC, which provides the following algorithms and properties. Here we give a high-level overview of the required properties and notation and refer the reader to figures in Appendix A for example algorithms.

AC must provide a commitment scheme, $\mathsf{AC.Commit}$, and algorithms such that the prover can convince a verifier that they know the value inside a commitment, which means that:

$$\mathrm{PK}\{(k, o) : c = \mathsf{AC.Commit}(k, o)\}.$$

We require the commitment scheme to be *perfectly hiding* and computationally binding, rather than the other way around. Indeed, we are more concerned with long-term privacy, which means that we are looking for information-theoretic security with respect to confidentiality. $\mathsf{AC.Commit}$ can be instantiated with the Pedersen commitment scheme [40], see Fig. 6.

AC must also contain a (blindable) signature scheme with the associated protocols enabling one to get a signature on a committed value ($\langle \mathsf{AC.GetSig}; \mathsf{AC.IssueSig} \rangle$) and to prove knowledge of a signature on a committed value ($\langle \mathsf{AC.ProveSig}; \mathsf{AC.VerifySig} \rangle$). This can be instantiated using CL-signatures [8], see Figs. 7 and 8.

The prover commits to a value k with commitment $c \leftarrow \mathsf{AC.Commit}(k, o)$ and opening o. Afterwards, they use $\sigma \leftarrow \langle \mathsf{AC.GetSig}; \mathsf{AC.IssueSig} \rangle$ to obtain a signature $\sigma = \mathsf{AC.Sign}(pk, sk, k, r)$ on the value k and some random value r. (pk and sk are the public verification key and the private signing key, respectively.)

At a later point, the prover wants to prove to a verifier that they know k and a signature σ on k made by the owner of pk (corresponding to sk) without revealing k nor σ (i.e., in a zero-knowledge manner). The prover and verifier run the protocol $\langle \mathsf{AC.ProveSig}; \mathsf{AC.VerifySig} \rangle$ to prove the following:

$$\mathrm{PK}\{(k, r) : \sigma' = \mathsf{AC.BlindSig}(\mathsf{AC.Sign}(pk, sk, k, r))\}.$$

Finally, we need a pseudo-random function (PRF), $\mathsf{AC.PRF}$, such that there exists a protocol $\langle \mathsf{AC.ProvePRF}; \mathsf{AC.VerifyPRF} \rangle$ implementing the following proof of knowledge (PK):

$$\mathrm{PK}\{(k) : y = \mathsf{AC.PRF}(k, x)\}.$$

This means that the prover can convince the verifier that $y = \mathsf{AC.PRF}(k, x)$ without revealing k. This can be instantiated by the Dodis and Yampolskiy [21] verifiable random function (VRF), see Fig. 9.

We need an anonymous credential system, AC, which provides the following algorithms and properties. Here we give a high-level overview of the required properties and notation and refer the reader to figures in Appendix A for example algorithms.

AC must provide a commitment scheme, AC.Commit, and algorithms such that the prover can convince a verifier that they know the value inside a commitment, which means that:

$$\mathrm{PK}\{(k, o) : c = \mathsf{AC.Commit}(k, o)\}.$$

We require the commitment scheme to be *perfectly hiding* and computationally binding, rather than the other way around. Indeed, we are more concerned with long-term privacy, which means that we are looking for information-theoretic security with respect to confidentiality. AC.Commit can be instantiated with the Pedersen commitment scheme [40], see Fig. 6.

AC must also contain a (blindable) signature scheme with the associated protocols enabling one to get a signature on a committed value ($\langle \mathsf{AC.GetSig}; \mathsf{AC.IssueSig} \rangle$) and to prove knowledge of a signature on a committed value ($\langle \mathsf{AC.ProveSig}; \mathsf{AC.VerifySig} \rangle$). This can be instantiated using CL-signatures [8], see Figs. 7 and 8.

The prover commits to a value k with commitment $c \leftarrow \mathsf{AC.Commit}(k, o)$ and opening o. Afterwards, they use $\sigma \leftarrow \langle \mathsf{AC.GetSig}; \mathsf{AC.IssueSig} \rangle$ to obtain a signature $\sigma = \mathsf{AC.Sign}(pk, sk, k, r)$ on the value k and some random value r. (pk and sk are the public verification key and the private signing key, respectively.)

At a later point, the prover wants to prove to a verifier that they know k and a signature σ on k made by the owner of pk (corresponding to sk) without revealing k nor σ (i.e., in a zero-knowledge manner). The prover and verifier run the protocol $\langle \mathsf{AC.ProveSig}; \mathsf{AC.VerifySig} \rangle$ to prove the following:

$$\mathrm{PK}\{(k, r) : \sigma' = \mathsf{AC.BlindSig}(\mathsf{AC.Sign}(pk, sk, k, r))\}.$$

Finally, we need a PRF, AC.PRF, such that there exists a protocol $\langle \mathsf{AC.ProvePRF}; \mathsf{AC.VerifyPRF} \rangle$ implementing the following PK:

$$\mathrm{PK}\{(k) : y = \mathsf{AC.PRF}(k, x)\}.$$

This means that the prover can convince the verifier that $y = \mathsf{AC.PRF}(k, x)$ without revealing k. This can be instantiated by the Dodis and Yampolskiy [21] VRF, see Fig. 9.

5.3 Distance-bounding protocols

Distance bounding (DB) [5] protocols were first suggested by Brands and Chaum [5] to prevent relay attacks in contactless communications in which the adversary forwards a communication between a prover and a possibly far-away verifier

to authenticate. These attacks cannot be prevented by cryptographic means as they are independent of the semantics of the messages exchanged. As a consequence, mechanisms ensuring the physical proximity between a verifier and a prover should be used instead. DB protocols precisely enable the verifier to estimate an upper bound on their distance to the prover by measuring the time-of-flight of short challenge-response messages (or rounds) exchanged during time-critical phases. Time critical phases are complemented by slow phases during which the time is not taking into account. At the end of a DB protocol, the verifier should be able to determine if the prover is legitimate *and* in their vicinity. In this sense, DB protocols combine the classical properties of authentication protocols with the possibility of verifying the physical proximity.

There are four adversaries for DB protocols established in the literature, each of which tries to commit a type of fraud. These can be summarized as follows:

- Distance fraud (DF) [5]: a legitimate but malicious prover wants to fool the verifier on the distance between them.
- Mafia fraud (MF) [19]: the adversary illegitimately authenticates using a, possibly honest, prover who is far away from the verifier. (Also known as relaying attack or man-in-the-middle attack.)
- Terrorist fraud (TF) [18]: a legitimate, but malicious, prover helps an accomplice, who is close to the verifier, to authenticate. TF resistance is a very strong property; it implies that if the verifier accepts the accomplice with non-negligible probability the accomplice can compute the prover's secret key[3].
- Distance hijacking (DH) [11]: similar to DF, the malicious prover is far away but uses an unsuspecting honest prover close to the verifier to pass as being close. (This is different from MF in that the honest prover actually tries to authenticate to the verifier, but the malicious prover hijacks the channel at some point(s) during the protocol.)

Our setting requires a public-key DB protocol with a *malicious verifier* who will potentially try to *impersonate the prover*. The verifier might also try to track the provers and map their identities to their actions, thus we also require strong privacy. In fact, as the construction in **??** shows, we require a DB zero-knowledge PK, or simply proof of proximity of knowledge (PPK) [46], for discrete logarithms. For this paper, we assume the existence of such a protocol. There exists PPK schemes, e.g. [46], just no published scheme for discrete logarithms yet[4].

[3]This means that even things like functional encryption will not help the adversary.
[4]The authors have such a protocol under submission in another venue.

5.4 Location proofs

Some location-based services only grant access to resources to users located at a particular location, thus raising the issue of verifying the position claimed by a particular user. One possible way to counter this threat is by having the requesting device formally prove that it really is at the claimed location, which gives rise to the concept of location proofs (LPs). In a nutshell, an LP is a digital certificate attesting that someone was at a particular location at a specific moment in time. A location-proof share (LPS) is an architecture by which users can obtain LPs from neighboring witnesses (e.g., trusted access points or other users) that can later be shown to verifiers who can check the validity of a particular proof [34, 50]. Most of the existing approaches to LPs require the prover and the witnesses to disclose their identities, thus raising many privacy issues such as the possibility of tracing the movements of users of the LPS. However, some LPSs, such as PROPS [27], exist that provide strong privacy guarantees along with the possibility of verifying the claim of the location.

Sonja ►*add something on platin.io, details unknown but roughly relying on witnesses and graph theory (unique big cluster, assumption of honest majority)*◄

5.5 Time-stamping and storage: ledger

We need a robust time-stamping and storage service (i.e., a ledger), L, which implements the following:

- $\rho \leftarrow \mathsf{L.GetStartPoint}$ yields a value ρ at time t such that ρ is difficult to guess before time t and $\mathsf{L.Time}(\rho) = t$;
- $\pi \leftarrow \mathsf{L.Submit}(x)$ stores x permanently and yields a value π at time t such that $\mathsf{L.Verify}(x, \pi) \rightarrow \top$ and $\mathsf{L.Time}(\pi) = t$.

With these building blocks, Alice can prove to a third-party verifier that a message m was created within the time interval $[t_0, t_1]$: After time t_0, Alice requests $\rho_{t_0} \leftarrow \mathsf{L.GetStartPoint}$. Before time t_1, Alice submits $h \leftarrow \mathsf{H}(m, \rho_{t_0})$ to L and gets $\pi_{t_1} \leftarrow \mathsf{L.Submit}(h)$. The tuple $(\rho_{t_0}, m, \pi_{t_1})$ can be used to prove that m was created within the time interval $[t_0, t_1]$. The verifier computes $h' \leftarrow \mathsf{H}(m, \rho_{t_0})$ and checks whether $\mathsf{L.Verify}(h', \pi_{t_1}) = \top$ and $\mathsf{L.Time}(\rho_{t_0}) \geq t_0 \wedge \mathsf{L.Time}(\pi_{t_1}) \leq t_1$.

The value output by $\mathsf{L.GetStartPoint}$ must be chosen at a low enough rate to not be unique for any individual. I.e., there must be a high probability that more than one person gets the same value from $\mathsf{L.GetStartPoint}$. (This can always be scaled, if $\mathsf{L.GetStartPoint}$ progresses at too high pace, one can resort to only using every nth output.)

L can be instantiated by an open-membership distributed ledger (e.g., a blockchain) such as Bitcoin [39], secured via Proof-of-Work consensus, or OmniLedger [32], secured via Byzantine consensus. If a blockchain is used for L, the $\mathsf{L.Submit}(x)$ request includes x in the blockchain and returns the identifier

of the block into which x was included. The L.GetStartPoint request returns the hash of the most recent block of the chain (i.e., the head).[5] The returned hash is difficult to predict since it depends on the content of the block, populated by other users and by the creator of the block with additional randomness (e.g., nonces and secrets).

Regarding consensus resilience, it is advisable to use a ledger with a high number of participants and preferably sharing the ledger with other services. We want to share the blockchain with other services due to privacy (and anti-censorship) reasons (analogous to the idea of "domain fronting").

We require a few additional properties from L that are already provided by ledgers. First, L must be continuously extended, such as in the Bitcoin blockchain in which blocks are created every 10 minutes on average (this is so that L.Time can map values from L.GetStartPoint to real time). Second, L must provide *immutability* and availability to any data committed through L.Submit to ensure verifiability of the data by anyone at any time.

6 Crowd counting with trusted witnesses

We now present a protocol for privacy-preserving but verifiable crowd counts. This version of the protocol relies on trusted entities to act as witnesses for an LP.

A prerequisite for this protocol is a one-to-one mapping between a persons real identity and a certificate (i.e., a cryptographic key). Section 6.1 covers how Alice can obtain such a certificate. It is a lighter version than in [35] and not the full protocol of [7].

At the core of our construction, witnesses generate participation proofs for the protesters (in Section 7.1), these are essentially augmented LPs[6]. Then whoever wants to verify the participation count, will count and verify those participation proofs (in Section 7.2).

The entities involved in our protocol are participants and (count) verifiers. A participant can assume three different roles:

(1) The *organizer* has written a manifesto for the protest and disseminated it to others. Anyone can do this.

(2) A *protester* is attending the protest and asks witnesses to vouch for their presence.

(3) A *witness* provides proofs of participation to protesters. The proofs are constructed such that they are verifiable by third parties. The witness must be trusted by the verifier.

[5] In situations where forks are common, it is relatively easy to adapt this process to look at a few blocks before the head and avoid the issue of the stamp becoming invalid later.

[6] An LP certifies that its owner was at a given location at a given time. In Section 3.1, we point out that we also need a cause identifier.

In general, there is one organizer and every participant can act as either or both protester and witness.

Anyone can be a verifier (we required universal verifiability, Section 2.1). The verifier defines the protest to be counted by setting the cause, time (interval) and location (area), cf., Definition 1. Now the verifier counts all proofs that verify correctly and fulfils the cause, time and location criteria. The verifier can publish, e.g., in a news paper, the final count together with the cause, time and location and anyone can verify this count (by the same procedure).

6.1 Prerequisite: self-certified, Sybil-free pseudonyms

Before Alice can have her participation in any event counted, she must get a certificate that ensures Sybil-freeness. This is only done once[7]. The keys can be reused for an arbitrary number of protests or, given careful choices in the PRF used for deriving identifiers, other services that work with anonymous credentials.

We use the setup and registration phases of Anon-Pass [33][8] for getting the certificate, adapting only the notation to fit ours.

Setup: $(spk_{CA}, ssk_{CA}) \leftarrow \mathsf{Setup}$ During the setup phase, the CA creates all the needed keys. The CA generates a service public-private key-pair $(spk_{CA}, ssk_{CA}) \leftarrow \mathsf{AC.Setup}$ (see Fig. 7).

Registration: $sk \leftarrow \langle \mathsf{Reg}_P(spk_{CA}); \mathsf{Reg}_{CA}(ssk_{CA}) \rangle$ During the registration phase, each participant generates a secret key (k, r) and obtains a signature on it by the CA *but without revealing it* to the CA (or to any part thereof in a decentralized CA scenario). At the end, each participant will have a signed secret key while the CA will issue only one signature per participant but without knowing the association between a particular key and the identity of the participant. The participant chooses $k, r \xleftarrow{\$} \mathbb{Z}_q$ uniformly randomly and runs $\sigma \leftarrow \langle \mathsf{AC.GetSig}(spk_{CA}, k, r); \mathsf{AC.IssueSig}(spk_{CA}, ssk_{CA}) \rangle$ (see Fig. 8). Upon success, the participant sets $sk = (\sigma, k, r)$.

6.2 Participation

The goal of our protocol is to generate and collect privacy-preserving participation proofs that can be counted and verified. These proofs consist of proof shares that are constructed as depicted in Fig. 2. They are constructed through an interactive protocol between the protester and the witness, as depicted in Fig. 3 and described below.

[7] It is repeated when the credential expires, in analogy to a passport in terms of expected intervals.

[8] This version is slightly adapted (improves efficiency) from the original version by Camenisch et al. [7].

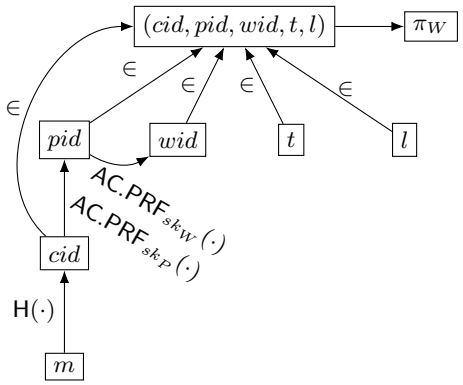

Figure 2: Structure of a proof share. The protest (cause) identifier *cid* is the hash value of the manifesto. The protester P's protest-specific pseudonym *pid* is computed using the protester's key sk_P and *cid*. The witness W's *protester*-specific identifier *wid* is computed using the witness's key sk_W and the protester's pseudonym *pid*. t is the time (interval) and l is the location (area) as determined by the trusted witness W. All values are signed by the witness (signature $\pi_W = \text{SPK}\{(sk_W) : wid = \cdots\}(cid, pid, wid, t, l)$) while also proving the correctness of *wid* and knowledge of a signature on sk_W by some CA.

Creation of a protest: the manifesto The organizer writes a manifesto for the protest, which describes its cause. This manifesto could take the form of any intelligible text, in practice at minimum a name. The organizer then distributes this manifesto to people through any suitable means (e.g., on the Web, on placards, etc.). If they agree with the cause, they will use the knowledge of the manifesto to join the protest.

Joining as a protester: $(pid) \leftarrow \mathsf{Join}_P(m)$ In the terminology of Martucci et al. [35], the manifesto m is the context and it yields an identity domain. A protester who wants to join the protest uses the manifesto m to compute an identifier for the cause (the context) by hashing the manifesto, $cid \leftarrow \mathsf{H}(m)$[9]. Afterwards, this is used to create the protester's protest-specific pseudonym $pid \leftarrow \mathsf{AC.PRF}_{sk_P}(cid)$.

Joining as a witness The witness does not have to do anything to join as a witness. That the witness is trusted by the verifier means that the verifier trusts that the witness can (1) determine the time of an interaction with a protester, (2) determine the its own location during that interaction and (3) will run the protocol as an honest witness with the potentially malicious protester.

[9]The result should be compared to that received from the organizer to check that the *cid* indeed is correct. This is to avoid that protesters use different *cid*s for semantically equivalent manifestos m. However, we omit this in the protocol for readability.

Participation: $\pi \leftarrow \langle \mathsf{Prticip}(cid, sk_P); \mathsf{Witness}(sk_W, spk_{CA}) \rangle$ In the participation phase, the protester and the witness construct the proof share of the protester (Fig. 2).

The protester sends pid to the witness. Then they run the protocol

$$\langle \mathsf{AC.ProveSig}(spk_{CA}, k, r, \sigma); \mathsf{AC.VerifySig}(spk_{CA}, ssk_{CA}) \rangle$$

(see Fig. 8), k and r are part of sk_P. Note that the proof of knowledge (PK) in Fig. 8 must be a proof of proximity of knowledge (PPK) [46] (i.e., a proof of knowledge with distance bounding). We use the protocol of **DB-Schnorr**, which does exactly this. If the protocol succeeds, the witness will compute $wid \leftarrow \mathsf{AC.PRF}_{sk_W}(pid)$ and send (wid, t, l) to the protester, where t is the current time and l is the witness' current location.

Submission: $s \leftarrow \mathsf{Submit}_W(cid, pid, wid, t, l)$ In the submission phase, the proof shares are be made available for the verifier. I.e., the verifier must be able to verify that a proof was actually issued by a witness.

To achieve this, the witness computes a non-interactive zero-knowledge (NIZK) proof ψ_{wid}, proving the correctness of wid while also signing the time t and location l. More specifically, we have that

$$\begin{aligned} \psi_{wid} \leftarrow \mathrm{SPK}\{(sk_W) : \\ & wid = \mathsf{AC.PRF}_{sk_W}(pid) \quad \wedge \\ & \sigma'_W = \mathsf{AC.BlindSig}\Big(\mathsf{AC.Sign}_{ssk_{CA'}}(sk_W)\Big)\Big\} \\ & \qquad\qquad\qquad\qquad\qquad\qquad (cid, pid, wid, t, l). \end{aligned}$$

Finally, the complete proof share is the tuple

$$s = (cid, pid, wid, t, l, \psi_{wid}).$$

For individual and universal verifiability, the proof should be published on some permanent storage, such as the ledger L by running $\mathsf{L.Submit}(s)$ (Section 5.5).

Note that it does not matter if it is the witness or the protester who makes s available to the verifier. The witness could compute s during the protest and give to the protester immediately, but we separate these steps to show that the witness can postpone those extra computations while potentially running on battery.

Now, the witness is anonymous here. For the verifier to recognize a proof on some publicly available storage as issued by a trusted witness there could be another CA, say CA', which issues credentials to witnesses. CA' could be run by a news paper, trusted to only issue credentials to trustworthy witnesses.

Another solution would be that the witness simply signs the tuple (cid, pid, t, l) with a key that is tied to the witness' identity. However, that would allow tracking and thus the anonymity set of the protester would shrink.

$$
\begin{aligned}
&O \to \text{all}\colon \text{manifesto}\\
&\qquad P\colon cid \leftarrow \mathsf{H}(\text{manifesto}),\\
&\qquad\quad pid \leftarrow \mathsf{AC.PRF}_{sk_P}(cid)
\end{aligned}
$$

Join

$$
\begin{aligned}
&P \to W\colon pid\\
&P \leftrightarrow W\colon \text{PPK}\{(sk_P)\colon\\
&\qquad\qquad pid = \mathsf{AC.PRF}_{sk_P}(cid),\\
&\qquad\qquad \sigma'_P = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}(sk_P)\big)\}\\
&\qquad W\colon wid \leftarrow \mathsf{AC.PRF}_{sk_W}(pid)
\end{aligned}
$$

Participation

(a) Join and participation

$$
\begin{aligned}
&W\colon \mathsf{L.Submit}((cid, pid, wid, t, l, \psi_{wid})), \quad \text{where}\\
&\quad \psi_{wid} = \text{SPK}\{(sk_W)\colon\\
&\qquad wid = \mathsf{AC.PRF}_{sk_W}(pid),\\
&\qquad \sigma'_W = \mathsf{AC.BlindSig}\Big(\mathsf{AC.Sign}_{ssk_{CA'}}(sk_W)\Big)\Big\}\\
&\qquad\quad (cid, pid, wid, t, l)
\end{aligned}
$$

(b) Submission

Figure 3: An overview of the protocol with trusted witnesses. The organizer O broadcasts the manifesto. The protester P with pseudonym pid in the context of the protest (cid), the witness W with pseudonym wid in the context of that protester and their computations are as in Fig. 2. Finally, W submits the proof share to a public ledger L for permanent storage.

Note also that the prover need not compute any NIZK proof of the correctness of pid since the witness is trusted to have verified this (as part of the distance bounding).

6.3 Count and verification

Assume that the verifier wants to verify the count for a protest $\mathcal{P}$. The first thing the verifier will do, is to download all the proof shares s from the ledger L, such that $s \sqsubseteq p$ is a valid proof share for some subprotest $p \in \mathcal{P}$ of the protest $\mathcal{P}$. Next, the verifier selects only those proof shares $s = (cid, pid, wid, t, l, \psi_{wid})$ such that ψ_{wid} proves knowledge of a signature by $ssk_{CA'}$. We denote this set of proof shares by $\mathcal{S}$.

Next, the verifier partitions the set of proof shares $\mathcal{S}$ using the relation $=_{pid}$ such that $(cid, pid, wid, t, l) =_{pid} (cid', pid', wid', t', l')$ is true if $pid = pid'$. Each equivalence class $\pi_{i,\mathcal{P}} \in \mathcal{S}/=_{pid}$ is a proof of participation for participant i. In

terms of Definition 4, we let $\theta = 1$ and

$$\varsigma(s) = \begin{cases} 1 & \text{if the verifier trusts } CA' \text{ to issue only to trustworthy witnesses} \\ 0 & \text{otherwise} \end{cases}.$$

Consequently, the total participation count is $|\Pi_{\mathcal{P}}^{\varsigma,\theta}|$.

7 Crowd counting with potentially untrusted witnesses

In line with some LPS, e.g., PRivacy-preserving lOcation-Proof System [27], we now provide a version where we reduce the trust in the witnesses. The entities and roles involved in this version of the protocol are still the same, but we reduce the possibility to create new proofs before and after an event.

7.1 Participation

The main change to participate is the use of the time-stamping property of L. Instead of the witness determining the time t, the witness and protester each determine a time interval for the creation of the proof. The witness gets $t_s \leftarrow \mathsf{L.GetStartPoint}$ and the protester gets $t'_s \leftarrow \mathsf{L.GetStartPoint}$ from the ledger L. They close the interval by committing the proof share to the ledger using $\mathsf{L.Submit}$. This adapted structure of a proof-share is depicted in Fig. 4. The updated protocol phases are given in Fig. 5 and described below.

The creation of the protest is the same: the organizer publishes the manifesto in some way.

Joining The join procedure is the same, except we add fetching t_s. So the protester computes $cid \leftarrow \mathsf{H}(m)$ and $pid \leftarrow \mathsf{AC.PRF}_{sk_P}(cid)$. The protester also fetches a time-correlated random value, t_s, from L, $t_s \leftarrow \mathsf{L.GetStartPoint}$.

The witness simply gets a time-correlated random value from the time-stamping service, $t'_s \leftarrow \mathsf{L.GetStartPoint}$. Note that we do this for redundancy, the newest of t_s and t'_s will set the start of the time interval of creation for the proof share.

Participation In the participation phase, the only difference is the use of t_s and t'_s instead of t.

The protester sends pid and t_s to the witness. Then they run the protocol

$$\langle \mathsf{AC.ProveSig}(spk_{CA}, k, r, \sigma); \mathsf{AC.VerifySig}(spk_{CA}, ssk_{CA}) \rangle$$

to verify the correctness of pid and do the distance bounding, same as before. If the protocol succeeds, the witness will compute $wid \leftarrow \mathsf{AC.PRF}_{sk_W}(pid)$ and send (wid, t'_s, l) to the protester.

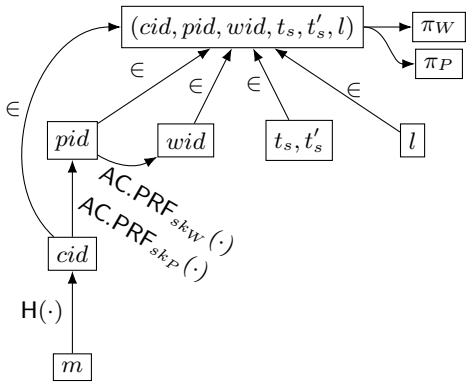

Figure 4: Structure of a proof share. The protest (cause) identifier *cid* is the hash value of the manifesto. The protester P's identifier *pid* is computed using the protester's key sk_P and *cid*. The witness W's protester-specific identifier *wid* is computed using the witness's key sk_W and the protester's *pid*. t_s, t'_s are the outputs from $\mathsf{L.GetStartPoint}$, e.g., hashes of the head blocks in the ledger seen by the protester and witness, respectively, and l is an area. All values are signed by the witness (signature $\pi_W = \mathrm{SPK}\{(sk_W) : wid = \cdots\}(cid, pid, wid, t_s, t'_s, l)$) while also proving the correctness of *wid* and knowledge of a signature on sk_W. The protester constructs π_P analogously.

Submission The submission phase differs in what constitutes the proof share. The protester commits the proof-share data to the ledger L and receives the proof of commitment $t_e \leftarrow \mathsf{L.Submit}(\mathsf{H}(cid, pid, wid, t_s, t'_s, l))$, which ends the time interval. The sooner this is done, the higher the precision for the time-dependent eligibility criterion will be for later counting. (The witness can also do this, the important part is to do it as soon as possible.) The remaining operations are not time critical.

The protester computes a NIZK proof ψ_{pid}, which shows the correctness of *pid* to a third party. More specifically,

$$\begin{aligned}\psi_{pid} \leftarrow \mathrm{SPK}\,\{(sk_P) :\ & \\ & pid = \mathsf{AC.PRF}_{sk_P}(cid) \quad \wedge \\ & \sigma'_P = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}(sk_P)\big)\big\} \\ & \qquad\qquad (cid, pid, wid, t_s, t'_s, l).\end{aligned}$$

Finally, the protester uploads the tuple

$$s_P = (cid, pid, wid, t_s, t'_s, t_e, l, \psi_{pid})$$

for permanent storage, $\mathsf{L.Submit}(s_P)$.

The witness, like the protester, commits the proof-share data to the ledger, $t_e \leftarrow \mathsf{L.Submit}(\mathsf{H}(cid, pid, wid, t_s, t'_s, l))$. (This is to close the time interval as

early as possible, whoever is the faster will submit it first, so both submit it.) **Sonja** ▶*but both do*◀ Then, without any time requirements, the witness computes a NIZK proof ψ_{wid} as follows:

$$\psi_{wid} \leftarrow \mathrm{SPK}\,\{(sk_W) : \\ wid = \mathsf{AC.PRF}_{sk_W}(pid) \quad \wedge \\ \sigma'_W = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}(sk_W)\big)\}\\ (cid, pid, wid, t_s, t'_s, l).$$

Finally, the witness uploads the tuple

$$s_W = (cid, pid, wid, t_s, t'_s, t'_e, l, \psi_{wid})$$

for permanent storage on the ledger, $\mathsf{L.Submit}(s_W)$.

7.2 Count and Verification

The set of proofs is constructed the same way as before. What differs is what we do with θ and ς.

We have the set of proof shares $\mathcal{S}$, as before. We also have the partitioning of the set of proof shares using the relation $=_{pid}$ such that $(cid, pid, wid, t, l) =_{pid} (cid', pid', wid', t', l')$ is true if $pid = pid'$. Each equivalence class $\pi_{i,\mathcal{P}} \in \mathcal{S}/=_{pid}$ is a proof of participation for participant i. This is the same as before.

However, in terms of Definition 4, the total participation count is still $|\Pi^{\varsigma,\theta}_{\mathcal{P}}|$, but this time the verifier might set θ and ς differently. The verifier can still set θ and ς the same as before and get the trusted witnesses scenario. But this time the verifier knows when the proofs were constructed (in $[t_s, t_e]$), due to trusting L to provide correct time-stamping and immutability.

Daniel ▶*The following should be edited to fit.*◀ Note that, thanks to the (ς, θ)-eligibility criterion (Definition 4), the method of counting is extremely generic, and each (counting) verifier can make an independent choice to regulate their trust in the final result, based on their initial trust in the witnesses. In other words, anyone who does the counting can choose the eligibility criteria (time interval, location, number of regular or trusted witnesses, who is considered to be a trusted witness) for their own count. As long as *these assumptions/criteria are published along with the result*, anyone can verify the correctness of the count under those criteria, and potentially question the validity of this choice. Biased or partisan verifiers may be tempted to make extreme choices, but they will have to publish those choices and lose credibility. Reasonable verifiers on the other hand will try to find a good middle-ground that counts all legitimate protesters while being resistant to isolated malicious agents.

$O \to \text{all}$: manifesto

P: $t_s \leftarrow \mathsf{L.GetStartPoint}$
$cid \leftarrow \mathsf{H}(\text{manifesto})$,
$pid \leftarrow \mathsf{AC.PRF}_{sk_P}(cid)$

W: $t'_s \leftarrow \mathsf{L.GetStartPoint}$

Join

Participation

$P \to W$: pid

$P \leftrightarrow W$: $\text{PPK}\{(sk_P) :$
$pid = \mathsf{AC.PRF}_{sk_P}(cid)$,
$\sigma'_P = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}(sk_P)\big)\}$

W: $wid \leftarrow \mathsf{AC.PRF}_{sk_W}(pid)$

$W \to P$: (wid, t'_s, l)

(a) Join and participation.

P: $t_e \leftarrow \mathsf{L.Submit}\big(\mathsf{H}\big(pid, wid, t_s, t'_s, l\big)\big)$

W: $t'_e \leftarrow \mathsf{L.Submit}\big(\mathsf{H}\big(pid, wid, t_s, t'_s, l\big)\big)$

W: $\mathsf{L.Submit}\big((cid, pid, wid, t_s, t'_s, t_e, l, \pi_{wid})\big)$, where
$\pi_{wid} = \text{SPK}\{(sk_W) :$
$wid = \mathsf{AC.PRF}_{sk_W}(pid)$,
$\sigma'_W = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}(sk_W)\big)\}$
$(cid, pid, wid, t_s, t'_s, l)$

P: $\mathsf{L.Submit}\big((cid, pid, wid, t_s, t'_s, t_e, l, \pi_{pid})\big)$, where
$\pi_{pid} = \text{SPK}\{(sk_P) :$
$pid = \mathsf{AC.PRF}_{sk_P}(cid)$,
$\sigma'_P = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}(sk_P)\big)\}$
$(cid, pid, wid, t_s, t'_s, l)$

(b) Submission.

Figure 5: An overview of CROCUS participation. The organizer O broadcasts the manifesto. The protester P, witness W and their computations are as in Fig. 4. Finally, both P and W submit the proof shares to a public ledger for permanent storage S. Note that *pid*always refers to the protester whose presence is being witnessed.

8 Security and privacy analysis

8.1 Eligibility verifiability

Requirement V1 states that anyone must be able to determine the authenticity of the relevant attributes of the data. In CROCUS, we have several attributes that must be verifiable: the time of creation (i.e., temporal eligibility, requirement V1.1), the physical location of sk_P at creation (i.e., spatial eligibility, requirement V1.2), recognition of two proofs originating from the same person (i.e., one-proof-per-person eligibility, requirement V1.3) and that the proof is indeed designated for the event (i.e., designated-event eligibility, requirement V1.4).

As we will show, it follows from Section 8.2 that the adversary cannot drop submitted proof shares and thus cannot decrease the count. As indicated in the adversary model (Section 3.3), the adversary can submit proof shares as everyone else, so the adversary's only option is to increase the count. We will thus let Alice pose as the adversary (malicious protester) in this section, as she naturally has an incentive to increase the count, as the organizer and a participant.

8.1.1 Temporal eligibility

Requirement V1.1 ensures freshness, as Alice cannot simply resubmit an old proof as a new one or create a proof in advance.

In general, to prevent replays, Alice must respond to an unpredictable challenge. The challenge here is "what did L.GetStartPoint return at the time of the proof's creation?". The response is included as t_s and t'_s in the proof share (see Fig. 4). The unpredictability of L.GetStartPoint ensures that a proof cannot have been created before $\max\{t_s, t'_s\}$. The correctness of the response must be verifiable by any verifier, which is the case with L.Time.

According to requirement V1.1, we must also prove that a proof share has not been created after a certain time. Otherwise, Grace could argue that the proof share was created after the protest, thus defeating the purpose of our protocol. The hash values of the proof shares are committed to the ledger (L.Submit), which means that there is a negligible probability that they were created after that: Alice would have to choose a value y in the range of the hash function H and then find a pre-image x such that $y = \mathsf{H}(x)$ and x is a valid proof for the desired protest, at the desired time. If H is collision resistant, she will succeed with negligible probability.

8.1.2 Counted only once

Requirement V1.3 aims to prevent Sybil attacks, in the sense that Alice cannot provide two (or more) participation proofs for a specific protest and thus be counted more than once. To do this she must create more than one pseudonym, pid. Indeed, to be counted twice, Alice must produce a $pid' \neq pid$. Due to the

deterministic property of AC.PRF, Alice must produce a new key sk'_P such that the verifier[10] accepts the proof

$$\mathrm{PK}\{(sk'_P) : pid' = \mathsf{AC.PRF}_{sk'_P}(cid) \quad \wedge \\ \sigma''_P = \mathsf{AC.BlindSig}\big(\mathsf{AC.Sign}_{ssk_{CA}}\big(sk'_P\big)\big)\}$$

while she does not know a valid signature on sk'_P. As a consequence, this is reduced to the security of the AC scheme. Remember, by assumption the CA will issue only one signature for such a key, so Alice cannot ask for a second one.

8.1.3 Spatial eligibility

Requirement V1.2 is achieved by having a witness vouch that Alice was indeed at the location when the proof share was created. In essence, the witness performs distance bounding to ensure Alice is close to them, this is then propagated to the verifier through the signature and founded on trust (remember that the witness is trusted by the verifier). Alice has three options: (1) relay her communication with an honest witness through a conspirator, (2) forge a witness signature for the proof, (3) corrupt a witness to issue a proof although neither might be present at the location.

Relaying the communication is an attack against the DB protocol. We assumed a DB protocol with TF resistance (the most relevant property for this situation), so Alice cannot succeed with more than negligible probability. (DF is captured by TF in this case, as in TF she has a colluder. MF would prevent Alice from relaying unsuspecting bystanders' communication to witnesses. DH is not really useful to Alice in this situation.)

Forging a witness's signature on a proof is equivalent to breaking the counted-only-once property above (Section 8.1.2). As above, this is reduced to the security of the AC scheme.

Finally, Alice can corrupt witnesses. In the case of trusted witnesses, Alice's chance of corrupting witnesses is reduced to the trustworthiness of the witnesses chosen by the verifier. In the θ-threshold case, with unknown witnesses, Alice succeeds only if she can corrupt at least θ witnesses.

We note that the strength function from Definition 4 allows the verifier to take different approaches, each of which must be individually analyzed. In all of these cases, it is up to the verifier to perform a risk analysis.

8.1.4 Designated use

Requirement V1.4 aims to prevent Alice (or someone else) from reusing the same proof (or proof share) for another event. This possibility is prevented through the use of *cid* in the proof shares. To reuse the proof share for another

[10] Here the verifier is either the witness during the distance bounding or the verifier who tries to verify the count.

protest, with a different manifesto, one must find a second pre-image m' such that $cid = \mathsf{H}(m) = \mathsf{H}(m')$.

There exists another case of collision that we must prevent. Consider the situation in which Alice computes $pid = \mathsf{AC.PRF}_{sk}(cid)$ for some cause identifier cid and some witnesses computes $wid_1, \ldots, wid_n$, with $wid_i = \mathsf{AC.PRF}_{sk_i}(pid)$. Now, if Alice constructs a manifesto m such that $\mathsf{H}(m) = pid$, then $wid_1, \ldots, wid_n$ would be valid participant identifiers for the protest with manifesto m. The protocol prevents such use by the fact that pid and wid are in fixed positions in both

$$\pi_{wid} = \mathrm{SPK}\{(sk_i) : \ldots\}(cid, pid, wid, t_s, t'_s, l)$$

and

$$\pi_{pid} = \mathrm{SPK}\{(sk_i) : \ldots\}(cid, pid, wid, t_s, t'_s, l),$$

and the two proofs can thus not be confused. Thus, the verification process differs for the two types of proofs.

8.2 Individual and universal verifiability

Requirement V3 means that Alice and Bob, as participants, can verify that their participation proofs (i.e., proof shares) are indeed included in the computed count. All proof shares (i.e., $\pi_{pid,\mathcal{P}} = (cid, pid, wid, t_s, t'_s, t_e, t'_e, l, \psi_{pid}, \psi_{wid})$) are committed to the ledger L and available from a public and permanent storage. Thus, Alice and Bob can simply check that all of their proof shares are indeed there and the security of individual verifiability depends on the properties of the ledger (cf., Section 5.5).

We assumed an honest-but-curious adversary controlling L[11]. This means that Alice can check that her proof share is indeed there.

Requirement V2 implies that anyone can check the result and that all participation proofs counted are legitimate. As the proof shares are committed and stored publicly, anyone can download them, verify eligibility (i.e., verify ψ_{pid}, ψ_{wid}) of the proofs and count them. Like for individual verifiability, the security of universal verifiability is reduced to the properties of the ledger; but universal verifiability also depends on the eligibility verification (each proof must be verified as eligible to count).

8.3 Privacy

The issue at core from a privacy perspective is linkability. We must ensure that no part of any proof is linkable to an individual.

We start with requirement P1. Given pid, the adversary cannot distinguish whether $pid = \mathsf{AC.PRF}_{sk_A}(cid)$ or $pid = \mathsf{AC.PRF}_{sk_B}(cid)$ due to the properties of AC (see [7]).

[11] We note that, in general, distributed (decentralized) ledgers cannot withstand a malicious Internet-service provider (ISP). Such an adversary can partition the network and provide Alice and Bob with different views of the ledger, thus breaking individual verifiability. However, this requires that the adversary *can observe* Alice's and Bob's channels to the ledger.

Requirement P2 means that Alice's proofs must be unlinkable across protests. This also follows from the properties of AC.PRF [7]: $pid = \mathsf{AC.PRF}_{sk}(cid)$ and $pid' = \mathsf{AC.PRF}_{sk}(cid')$ (where $cid \neq cid'$) are unlinkable from the perspective of the adversary. The argument is the same for requirement P3.

However, there are more data than pid, wid used in the protocol. The protocol uses $cid, pid, wid, t_s, t'_s, l, \psi_{pid}, \psi_{wid}$. The cause identifier cid will be used for all proof shares pertaining to the same protest and thus it cannot be used to uniquely identify any individual. The location l is coarse enough so that many non-overlapping (pid, wid)-pairs use the same location. After all, it is the location of the protest, not the location within the protest that matters. Thus l is not uniquely identifying any individual protester or witness. Likewise, thanks to the constraints in the time-stamp granularity (Section 5.5), t is not uniquely identifying either.

9 Performance

Sonja ►*include evaluation environment, parameters, results, disclaimer about parts that cannot yet be implemented and evaluated because the hardware doesn't exist yet for DB on phones*◄

Performance considerations are crucial during the protests due to the nature of the devices used to run CROCUS, which are resource-constrained in terms of energy, storage and computational power, and are operating on limited network capacity.

9.1 Smartphones and smartcards

Recent technological progress has enabled the deployment of advanced cryptographic primitives on smartcards and smartphones that could be used to implement our solution. For instance, benchmarks [29] have shown that Android devices are now fast enough to efficiently implement privacy-enhancing technologies, with a Samsung Galaxy S i9000 (back in 2014) able to execute Idemix in 153 ms. However, those benchmarks also demonstrate that smartcards remain slow to process complex protocols such as Idemix or U-Prove (taking between 4 s and 8 s to process them). While the limited processing power of many embarked systems has been a challenge, Idemix has been successfully implemented to prove the possession of credentials on Java Cards by Bichsel and co-authors in 2009 [4] and the IRMA project, released in 2014, aimed to achieve an implementation "suitable for real life transactions" [16] while maintaining security and privacy for its users.

With respect to its implementation, CROCUS is very similar to the implementation of Anon-Pass [33], an anonymous subscription system in which a long-term credential can be used to derive a single login for any authentication window (i.e., epoch) such that logins are unlinkable across different epochs. In particular, the setup and registration phases are almost identical. The evaluation conducted by Lee et al. [33] used as server a Dell Optiplex 780s, which has

a quad-core 2.66 GHz Intel Core 2 CPU, 8 GB of RAM and uses Ubuntu Linux 12.04 while the client was also simulated on a quad-core 2.66 GHz Intel Core 2 CPU but with only 4 GB of RAM. The elliptic pairing group used in Anon-Pass is a Type A symmetric pairing group with a 160-bit group order and 512-bit base field while the ECDSA signature uses a 160-bit key. This is typically the type of pairing that could also be used with CROCUS . Not counting the setup that is performed once by the CA, the time reported for the registration on the server (i.e., CA) side is 19.8 ms while it is 23.4 ms on the client side. With respect to joining and participation phases, with the exception of the distance-bounding they are actually quite similar to the login protocol of Anon-Pass in which the time required to create a message for a protester would be around 13.5 ms while the time needed for the witness would be only 7.9 ms.

9.2 Witness processing

As above, assume we have one trusted witness. A smartphone ran Idemix in 153 ms back in 2014. 3DB-access [1] says their distance-bounding part takes less than 1 ms and can do more than 120 m (line of sight, maximum 200 m). Then a witness fixed in one position can cover 45 216 m^2 (120 m radius), which fits 137018 protesters according to Tong-Hyung and Lee [44]. It would take this witness 5.8 h to witness all of them. Thus, to witness 1 000 000 protesters, it would take 8 h for 5.3 witnesses.

9.3 Ledger (blockchain) efficiency

For the example of 1 000 000 participants, with trusted witnesses, each participant only needs to acquire one proof share from a trusted witness. Thus, there will be 1 000 000 proof shares submitted to the blockchain in total. If we consider OmniLedger [32], which can do approximately 1500 transactions per second, it takes at least 11 minutes to process all the proof shares.

Sonja ►*removed for now: One problem can arise in the case in which there is no trusted witnesses. In this situation, if we set the threshold too high, we need to submit more proof shares than the blockchain can handle within a reasonable time.*◄

10 Related work

Our scheme relates to several other works: the crowd counting methods already described in Section 2, location-proof systems and Sybil-free pseudonym systems.

Location-proof systems In a nutshell, an LP is a digital certificate attesting that someone was at a particular location at a specific moment in time. An LPS is an architecture by which users can obtain LPs from neighboring witnesses (e.g., trusted access points or other users) that can later be shown to

verifiers who can check the validity of a particular proof [34, 50]. Most of the existing approaches to LPs require the prover and the witnesses to disclose their identities, thus raising many privacy issues such as the possibility of tracing the movements of users of the LPS. However, some LPSs, such as PROPS [27], exist that provide strong privacy guarantees along with the possibility of verifying the claim of the location.

CROCUS shares some similarities with PROPS. The main difference is that CROCUS operates in a more adverse environment, and can thus not provide any guarantees using the decentralized, untrusted witnesses that PROPS can. The incentives to cheat are also bigger and consequently the thresholds for collusion are much higher. CROCUS must also tie the cause to the location proof, to designate the proof to the protest in which the protester participated.

The same can be said in comparison to Platin.io[12].

Sybil-free pseudonym systems Our Sybil-free pseudonym system is very similar to that of Martucci et al. [36]. Both are based on the work of Camenisch et al. [7]. We make the same simplification of [7] as Martucci et al. [36], but we also require an interactive version. Furthermore, our PKs must be distance bounding.

11 Discussion

11.1 Implementability

Some of the assumptions that are required for implementing our proposition are not yet realized, however, we believe that they soon will be.

One major issue is that all protesters must have smartphones — in many countries with oppressive regimes, far from all possess smartphones currently, though the rate is on the rise.

On those smartphones we additionally require an identity credential signed by some CA. This is also not yet widely available. However, more and more nation states are starting to issue digital certificates in identity cards and many already have crypto-enabled RFID chips in their passports. E.g., Estonia, Germany, and Sweden already have the infrastructure and widely deployed electronic identity systems, and the EU already has regulation in place (eIDAS). In Sweden, more than 95 % of people in the ages 21–50 use BankID, 88 % for ages 51–60 and 76 % for ages 61–70 [48].

National identities, however, are problematic when there is an incentive for the government to create more identities, for example when it comes to elections. In functioning voting systems, there is no more than one ballot (token) per physical person and the ballot is not linked to the vote. Audits and other processes help ensure that there are no ballots for non existing persons (Sybil-proof identities) or extra ballots for voters. Similarly, here, we need (1) a mapping between an identity and a physical person (in the form of an anonymous

[12]URL: `https://platin.io`.

identity credential, functioning as a pseudonym for privacy properties) and (2) only one such credential per identity (token of physical personhood instead of token of being a voter in a particular election). With the same caveat as for identities for voting for (1), mechanisms such as collective signing [42] could ensure (2). Again, this is not yet realized, but with the current pace of adopting public ledgers and efforts for cross-national identity credentials such as the EU's eIDAS, the prerequisites exist. Moreover, the code of practice for European statistics [24] includes principles such as coordination and cooperation as well as impartiality and objectivity and is an example of both efforts toward and motivation for cross-national improvements of statistics.

We also need to run distance-bounding protocols on smartphones. Achieving this is currently not feasible within a meaningful range as existing smartphones lack the required hardware to conduct the distance bounding fast enough. However, thefts of luxury cars due to relay attacks have driven the development of hardware for doing distance bounding in car keys. We believe that using smartphones for contactless payment and electronic tickets will drive a similar development for this hardware on smartphones.

If we remove the distance-bounding component, we can, given some caveats on trust and setup, use CROCUS for online petitions: a trusted consortium could set up a petition site, ensure that only one credential is issued per identity (given sybil-proof identities, again, building on something like eIDAS), use CROCUS without witnesses and location and still benefit from the privacy and verifiability properties from its pseudonyms and ledger setup.

11.2 Communications

We need the participants to communicate during the protest. Since the cellular network could be shut down to keep protesters from accessing the Internet, making phone calls or texting, we require a different means of communications between protesters. This could be accomplished by Bluetooth or WiFi communications, as demonstrated by Briar [6] and FireChat [26], two examples of applications for communication during protests via wireless mesh networking. The crowd-counting scenario likely has higher requirements on capacity and withstanding interference as the participants continuously run the protocol for witnessing each others presence; messages are presumably less frequent. 5G is intended to cope with billions of devices (IoT), and thus could help cope with the device density in crowds. An alternative, although originally designed to work within 5G cellular networks, is device-to-device communication (D2D) [31]. Specifically, the out-of-band and autonomous version D2D would fit our scenario thanks to using unlicensed spectrum and working without cellular coverage. Due to the requirements of lawful intercept and the drive for operators to identify the network users, there still is an authorization step to communicate in this mode. Near-field-communication (NFC) would solve any scalability problem from interference, require no infrastructure or provider, but at the price of having participant and witness hold their phones together for each proof share.

11.3 Adversaries

Our adversary model considers only protocol data, no auxiliary data. Against this adversary our scheme is secure. The question is how this adversary model maps to real adversaries.

In any real implementation there are potential side-channels. E.g., in the communication layer: IP-addresses translate into identities, devices' MAC-addresses can be used as persistent identifiers. We do not consider these aspects as there are entire fields dedicated to some of them, we simply assume the tools developed in those fields (e.g., Tor [20] and randomized MAC-addresses) to prevent these problems will be used.

Like other schemes involving Internet communication, we do not consider a global passive adversary. Such an adversary could use side information to do e.g., time-correlation attacks against people that submit transactions concerning a particular *cid* over Tor, identify them, and link them to the *cid*. On a more realistic scale, a national passive adversary can control all the nation's ISPs but would not be able to observe all Tor exit nodes or otherwise observe all input to the ledger needed to perform such correlation attacks.

During a protest there are also other information channels available to the adversary. E.g., one could argue that the adversary might be able to map a face to a *pid* by means of signal triangulation during the protocol run, and then map the face to an identity through face recognition. However, there are far easier tactics the adversary could use: e.g., the adversary can take photos of the event and try to capture as many faces as possible. This is already possible today and thus not a weakness introduced by our scheme.

Besides these privacy concerns, there are also verifiability concerns. There, we have shown that the security is reduced to the witnesses. In the case of using trusted witnesses — which we consider the canonical CROCUS— it simply reduces to their trustworthiness, like in any system with trusted third parties.

We outlined a variant of counting and verifying for CROCUS that uses a threshold θ of untrusted witnesses and, trivially, resists a collusion of malicious witnesses smaller than θ. While using this approach is technically possible, we do not know whether it is of practical use. We have not found a principled way of determining a secure value for θ, yet neither can we at this point conclude that there is no such way. Informally, a higher θ increases the probability of a collusion that is large enough to break verifiability being detected and made known to the verifier.

Overall, while CROCUS provides the mechanism for a privacy-preserving and verifiable way of counting crowds, any verifier still needs to consider their own trust, analogous to detecting bias in science. There, if a paper espousing, say, the efficacy of X, comes from research sponsored by the company that manufactures X, it might be biased. Likewise, if a count of a pro-government protest is published by that government, one needs to consider where identities were issued and which witnesses were considered trusted. In contrast to a layperson interpreting science, however, the results can easily be reproduced by the verifier: knowing which set of witnesses (and other criteria such as lo-

cation and time) the counter used for the count, the count can be verified (by a re-count). Whether the result can be trusted then depends on whether the verifier also trusts this set of witnesses. Given that all proof shares are on the ledger, however, a verifier can come up with their own criteria and make a count themselves. This count then, given that they publish what criteria they used, can in turn be verified by anyone else.

12 Conclusion

In this paper, we have introduced CROCUS, a privacy-preserving protocol for verifiably counting participants at protests.

We showed that CROCUS provides universally verifiable data. With CROCUS, a journalist can easily count the participation, specify how they counted (time, location etc.) along with the result and everyone can independently verify that the result is correct. The only results that cannot be trusted are results aligned with the interests of the CA (which can be mitigated by multi-party solutions).

Despite the verification, the privacy of the participants is preserved to the extent possible by their physical presence at the protest, with the following caveat. Grace, the government in our protest scenario, can coerce Alice in some way to reveal her private key and then use it to verify her participation. However, this requires that Grace already suspects Alice — Grace cannot check everyone — and that Alice has not renewed her digital certificate.

While CROCUS is an actual count and not an estimate, its accuracy for the total number of participants hinges on the participants having the necessary equipment (i.e., a smartphone or similar device), some sort of trustworthy credential, and the willingness to run the protocol. Until used by most participants, CROCUS would result in a considerable undercount. However, it represents a first step toward accurate verifiable yet privacy-preserving crowd counting by showing how it can be done in theory at least and in practice once some assumptions concerning hardware and e-identities become more realistic. While the necessary technologies are not yet available in most authoritarian regimes, some of them are in several democracies. We believe that it is important to implement systems such as ours also there to support the maintenance of democratic processes and increase transparency.

References

[1] 3db Access AG. *3db Access*. URL: https://www.3db-access.com.

[2] Farouk El-Baz. *The [?]-Man March*. WIRED. June 2003. URL: https://www.wired.com/2003/06/crowd-spc/ (visited on 07/21/2017).

[3] BBC Magazine. "Protest numbers: How are they counted?" In: *BBC News* (Mar. 28, 2011). URL: http://www.bbc.com/news/magazine-12879582 (visited on 07/24/2017).

[4] Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Tom Heydt-Benjamin, Dieter Sommer, and Greg Zaverucha. "Cryptographic protocols of the identity mixer library". In: *Tech. Rep. RZ 3730, Tech. Rep.* (2009).

[5] Stefan Brands and David Chaum. "Distance-bounding protocols". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1993, pp. 344–359.

[6] Briar. *Briar: a messaging app designed for activists*. 2016. URL: https://briarproject.org/how-it-works.html (visited on 02/25/2018).

[7] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. "How to Win the Clonewars: Efficient Periodic N-times Anonymous Authentication". In: *13th ACM Conference on Computer and Communications Security*. 2006, pp. 201–210. DOI: 10.1145/1180405.1180431.

[8] Jan Camenisch and Anna Lysyanskaya. "Signature schemes and anonymous credentials from bilinear maps". In: *Annual International Cryptology Conference*. Springer. 2004, pp. 56–72.

[9] Jan Camenisch and Markus Stadler. "Efficient group signature schemes for large groups". In: *Annual International Cryptology Conference, CRYPTO'97*. Springer. 1997, pp. 410–424.

[10] Nicolas Chapuis. "Des médias s'associent pour compter les participants aux manifestations". fr. In: *Le Monde.fr* (Mar. 2018). ISSN: 1950-6244. URL: https://www.lemonde.fr/societe/article/2018/03/20/des-medias-s-associent-pour-compter-les-participants-aux-manifestations_5273676_3224.html (visited on 05/30/2018).

[11] Cas Cremers, Kasper B Rasmussen, Benedikt Schmidt, and Srdjan Capkun. "Distance hijacking attacks on distance bounding protocols". In: *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE. 2012, pp. 113–127.

[12] CrowdCount. *Crowd Count: Real-time crowd sizing*. 2016. URL: http://crowdcount.org/ (visited on 04/05/2017).

[13] CrowdSize. *Crowdsize iPhone Application*. 2016. URL: http://www.crowdsize.com/ (visited on 07/24/2017).

[14] Peter Danielis, Sylvia T Kouyoumdjieva, and Gunnar Karlsson. "DiVote: A Distributed Voting Protocol for Mobile Device-to-Device Communication". In: *Teletraffic Congress (ITC 28), 2016 28th International*. Vol. 1. IEEE. 2016, pp. 69–77.

[15] Peter Danielis, Sylvia T Kouyoumdjieva, and Gunnar Karlsson. "UrbanCount: Mobile Crowd Counting in Urban Environments". In: *8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), October 03-05, 2017, Univ British Columbia, Vancouver, Canada*. Institute of Electrical and Electronics Engineers (IEEE). 2017, pp. 640–648.

[16] Antonio De La Piedra, Jaap-Henk Hoepman, and Pim Vullers. "Towards a full-featured implementation of attribute based credentials on smart cards". In: *International Conference on Cryptology and Network Security*, pp. 270–289.

[17] Stéphanie Delaune, Steve Kremer, and Mark Ryan. "Verifying privacy-type properties of electronic voting protocols". In: *Journal of Computer Security* 17.4 (2009), pp. 435–487.

[18] Yvo Desmedt. "Major security problems with the 'unforgeable'(feige)-fiat-shamir proofs of identity and how to overcome them". In: *Proceedings of SECURICOM*. Vol. 88. 1988, pp. 15–17.

[19] Yvo Desmedt, Claude Goutier, and Samy Bengio. "Special uses and abuses of the Fiat-Shamir passport protocol". In: *Conference on the Theory and Application of Cryptographic Techniques*. Springer. 1987, pp. 21–39.

[20] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. "Tor: The Second-Generation Onion Router". In: *USENIX Security Symposium*. 2004, pp. 303–320.

[21] Yevgeniy Dodis and Aleksandr Yampolskiy. "A verifiable random function with short proofs and keys". In: *International Workshop on Public Key Cryptography*. Springer. 2005, pp. 416–431.

[22] John R. Douceur. "The Sybil Attack". In: *Peer-to-Peer Systems*. 2002. DOI: 10.1007/3-540-45748-8_24.

[23] Sam Edwards. *Barcelona protesters demand release of jailed separatist leaders*. [Online; accessed 2. Feb. 2018]. Nov. 11, 2017. URL: https://www.independent.co.uk/news/world/europe/barcelona-protesters-demand-release-of-jailed-separatist-leaders-catalonia-latest-a8050116.html (visited on 02/02/2018).

[24] European Statistical System Committee. Nov. 2017. DOI: 10.2785/798269. URL: http://ec.europa.eu/eurostat/web/quality/european-statistics-code-of-practice.

[25] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO' 86: Proceedings*. 1987. DOI: 10.1007/3-540-47721-7_12.

[26] FireChat. *FireChat: a messaging app without Internet access or cellular data*. 2016. URL: https://www.opengarden.com/firechat.html (visited on 02/25/2018).

[27] S. Gambs, M.-O. Killijian, M. Roy, and M. Traore. "PROPS: A PRivacy-preserving lOcation Proof System". In: *Reliable Distributed Systems (SRDS), 2014 IEEE 33rd International Symposium on*. 2014. DOI: 10.1109/SRDS.2014.37.

[28] Michelle Goldberg. *The protest-crowd numbers game - Salon.com*. Salon. Jan. 2003. URL: http://www.salon.com/2003/01/24/crowds/ (visited on 07/21/2017).

[29] Jan Hajny, Lukas Malina, Zdenek Martinasek, and Ondrej Tethal. "Performance evaluation of primitives for privacy-enhancing cryptography on current smart-cards and smart-phones". In: *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2014, pp. 17–33.

[30] Al-Jazeera. *Huge marches as Venezuela marks 50 days of protest*. May 21, 2017. URL: http://www.aljazeera.com/news/2017/05/venezuelan-opposition-marks-50-days-protests-170520174956348.html (visited on 08/01/2017).

[31] Udit Narayana Kar and Debarshi Kumar Sanyal. "An overview of device-to-device communication in cellular networks". In: *ICT Express* (2017). ISSN: 2405-9595. DOI: 10.1016/j.icte.2017.08.002.

[32] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. *OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding*. Cryptology ePrint Archive, Report 2017/406. To appear in 39th IEEE S&P 2018. 2017.

[33] Michael Z Lee, Alan M Dunn, Brent Waters, Emmett Witchel, and Jonathan Katz. "Anon-pass: Practical anonymous subscriptions". In: *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE. 2013, pp. 319–333.

[34] Wanying Luo and Urs Hengartner. "Veriplace: a privacy-aware location proof architecture". In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2010, pp. 23–32.

[35] Leonardo A. Martucci, Markulf Kohlweiss, Christer Andersson, and Andriy Panchenko. "Self-Certified Sybil-Free Pseudonyms". In: *Proceedings of the First ACM Conference on Wireless Network Security*. WiSec '08. Alexandria, VA, USA: Association for Computing Machinery, 2008, pp. 154–159. ISBN: 9781595938145. DOI: 10.1145/1352533.1352558. URL: https://doi.org/10.1145/1352533.1352558.

[36] Leonardo A. Martucci, Markulf Kohlweiss, Christer Andersson, and Andriy Panchenko. "Self-certified Sybil-free Pseudonyms". In: *Proceedings of the First ACM Conference on Wireless Network Security*. WiSec '08. Alexandria, VA, USA: ACM, 2008, pp. 154–159. ISBN: 978-1-59593-814-5. DOI: 10.1145/1352533.1352558.

[37] Remy Melina. *How Is Crowd Size Estimated?* Live Science. Sept. 2010. URL: https://www.livescience.com/8578-crowd-size-estimated.html (visited on 07/20/2017).

[38] Robinson Meyer. "How Will We Know Trump's Inaugural Crowd Size?" In: *The Atlantic* (Jan. 20, 2017). ISSN: 1072-7825. URL: https://www.theatlantic.com/technology/archive/2017/01/how-will-we-know-trumps-inaugural-crowd-size/513938/ (visited on 07/24/2017).

[39] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[40] Torben Pryds Pedersen. "Non-interactive and information-theoretic secure verifiable secret sharing". In: *Annual International Cryptology Conference*. 1991, pp. 129–140.

[41] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. v0.34. Aug. 2010. URL: http://dud.inf.tu-dresden.de/literatur/Anon%5C_Terminology%5C_v0.34.pdf.

[42] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. "Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning". In: *2016 IEEE Symposium on Security and Privacy (SP)*. May 2016, pp. 526–545. DOI: 10.1109/SP.2016.38.

[43] Alan Taylor. *Months of Deadly Anti-Government Protests in Venezuela - The Atlantic*. June 12, 2017. URL: https://www.theatlantic.com/photo/2017/06/months-of-anti-government-protests-continue-in-venezuela/530031/ (visited on 08/01/2017).

[44] Kim Tong-Hyung and Youkyung Lee. "Counting 1 million crowds at anti-president rallies in Seoul". In: *Associated Press: The Big Story* (Nov. 2016). URL: http://bigstory.ap.org/article/317ea62bddbd4132ab1467863a532ab9/counting-1-million-crowds-anti-president-rallies-seoul (visited on 04/05/2017).

[45] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. "Why MAC Address Randomization is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms". In: *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security*. ASIA CCS '16. New York, NY, USA: ACM, 2016, pp. 413–424. ISBN: 978-1-4503-4233-9. DOI: 10.1145/2897845.2897883. URL: http://doi.acm.org/10.1145/2897845.2897883 (visited on 07/24/2017).

[46] Serge Vaudenay. "Sound proof of proximity of knowledge". In: *International Conference on Provable Security*. Springer. 2015, pp. 105–126.

function $\mathsf{P.Commit}(x, r)$
  return $g^x h^r$

Figure 6: Pedersen's commitment scheme [40]. Let $G = \langle g \rangle = \langle h \rangle$ be a group with prime order q and generators g and h.The r should be chosen randomly from $\mathbb{Z}_q$.

[47] Kaveh Waddell. "The Exhausting Work of Tallying America's Largest Protest". In: *The Atlantic* (Jan. 2017). ISSN: 1072-7825. URL: https://www.theatlantic.com/technology/archive/2017/01/womens-march-protest-count/514166/ (visited on 04/05/2017).

[48] Malin Wemnell. *Marknadsinformation BankID — användning och innehav*. Swedish. English title: Market information BankID — use and holding. Apr. 2018. URL: https://www.bankid.com/assets/bankid/stats/2018/statistik-2018-04.pdf.

[49] Cong Zhang, Hongsheng Li, X. Wang, and Xiaokang Yang. "Cross-scene crowd counting via deep convolutional neural networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 833–841. DOI: 10.1109/CVPR.2015.7298684.

[50] Zhichao Zhu and Guohong Cao. "Applaus: A privacy-preserving location proof updating system for location-based services". In: *INFOCOM, 2011 Proceedings IEEE*. IEEE. 2011, pp. 1889–1897.

A Anonymous credentials protocol details

Here we summarize the algorithms suggested as instantiations for the anonymous-credentials system AC in Section 5.2: these are summarized as Figs. 6 to 10.

function CL.Setup
  $x \xleftarrow{\$} \mathbb{Z}_q, X \leftarrow g^x, y \xleftarrow{\$} \mathbb{Z}_q, Y \leftarrow g^y, z \xleftarrow{\$} \mathbb{Z}_q, Z \leftarrow g^z$
  $sk \leftarrow (x, y, z), pk \leftarrow (q, G, G_T, g, g_T, e, X, Y, Z)$
  return (sk, pk)

function CL.Sign(pk, sk, m, r)
  $a \xleftarrow{\$} G, A \leftarrow a^z$
  $b \leftarrow a^y, B \leftarrow A^y$
  $c \leftarrow a^{x+xym} A^{xyr}$
  return $\sigma = (a, A, b, B, c)$

function CL.BlindSig$(\sigma = (a, A, b, B, c))$
  $r \xleftarrow{\$} \mathbb{Z}_q, r' \xleftarrow{\$} \mathbb{Z}_q$
  $\tilde{a} \leftarrow a^r, \tilde{A} \leftarrow A^r, \tilde{b} \leftarrow b^r, \tilde{B} \leftarrow B^r, \hat{c} \leftarrow (c^r)^{r'}$
  return $\tilde{\sigma} = (\tilde{a}, \tilde{A}, \tilde{b}, \tilde{B}, \hat{c})$

function CL.VerifySig$(pk, m, r, \sigma = (a, A, b, B, c))$
  if $e(a, Z) \neq e(g, A)$ **then**
    return $\bot$ ▷ A malformed
  else if $e(a, Y) \neq e(g, b) \vee e(A, Y) \neq e(g, B)$ **then**
    return $\bot$ ▷ b or B malformed
  else if $e(X, a) \cdot e(X, b)^m \cdot e(X, B)^r \neq e(g, c)$ **then**
    return $\bot$ ▷ c malformed
  return $\top$

Figure 7: The CL-signature scheme [8]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \to G_T$ be a bilinear map.

CL.GetSig(pk, m, r)		CL.IssueSig(pk, sk)
$M \leftarrow$ P.Commit(m, r)	$\xrightarrow{M}$	
	PK$\{(m, r) : M =$ P.Commit$(m, r)\}$	
		$\alpha \xleftarrow{\$} \mathbb{Z}_q$
		$a \leftarrow g^\alpha, A \leftarrow a^z$
		$b \leftarrow a^y, B \leftarrow A^y$
$\sigma \leftarrow (a, A, b, B, c)$	$\xleftarrow{(a,A,b,B,c)}$	$c \leftarrow a^x M^{\alpha xy}$

CL.ProveSig(pk, m, r, σ)		CL.VerifySig(pk, sk)
$\tilde{\sigma} \leftarrow$ CL.BlindSig(σ)	$\xrightarrow{\tilde{\sigma}}$	$e(\tilde{a}, Z) \stackrel{?}{=} e(g, \tilde{A})$
		$e(\tilde{a}, Y) \stackrel{?}{=} e(g, b)$
		$e(\tilde{A}, Y) \stackrel{?}{=} e(g, \tilde{B})$
$v_x \leftarrow e(X, \tilde{a})$		$v_x \leftarrow e(X, \tilde{a})$
$v_{xy} \leftarrow e(X, \tilde{b})$		$v_{xy} \leftarrow e(X, \tilde{b})$
$v_s \leftarrow e(g, \hat{c})$		$v_s \leftarrow e(g, \hat{c})$
	PK$\{(m, r) : v_s^r = v_x v_{xy}^m\}$	

Figure 8: Protocols for CL anonymous credentials [8]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \to G_T$ be a bilinear map.

function DY.SetupPRF
  $sk \xleftarrow{\$} \mathbb{Z}_q^*$
  $pk \leftarrow g^{sk}$
  return (sk, pk)

function DY.PRF(sk, x)
  return $y = g_T^{\frac{1}{sk+x}}$

function DY.ProvePRF(sk, x)
  return $\pi = g^{\frac{1}{sk+x}}$

function DY.VerifyPRF(pk, x, y, π)
  if $e(g^x \cdot pk, \pi) \neq e(g, g)$ **then**
    return $\bot$
  else if $y \neq e(g, \pi)$ **then**
    return $\bot$
  return $\top$

Figure 9: Verifiable random function [21]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \to G_T$ be a bilinear map.

CHKL.ProvePRF(k, x)	CHKL.VerifyPRF(y)
$y \leftarrow$ DY.PRF(k, x)	

$$\mathrm{PK}\{(k) : y = \mathsf{DY.PRF}(k, x)\}$$

Figure 10: Protocols using DY.PRF with CL anonymous credentials [7]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \to G_T$ be a bilinear map.