sciendo

From grassroots to CROCUSes: privacy-preserving CROwd Counting Using Smartphones

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.

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). 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. Bob, Carol and many others participate in the protest. However, they might only be willing to do so in a manner that does not leave any digital traces because of the fear 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.

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 [42]. 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 [42], Trump's inauguration [37], the 2017 Women's March in the US [45], the demonstrations against the change of constitution in Venezuela [31] or for the independence of Catalonia [24].

Existing methods for crowd-counting vary significantly in terms of approaches (we will review them in detail in Section 3). However, most of them lack precision (i.e., they have large error margins) and they can only give an estimate for a particular snapshot in time. In particular, the existing methods cannot reliably estimate the cumulative participation count — at least not without counting some persons multiple times — which in turn increases the error of the estimation. 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. Indeed, going back to our example, 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 case, we are left with two options: either we trust Alice or Grace (or another third party) or we have to be able to verify their claims ourselves. 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.

2 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.

2.1 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 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 Section 6 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. This assumption is necessary to ensure that the participants will be able to upload their proof shares to the ledger.

2.2 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]:

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

Universal verifiability: anyone can verify that the result is according to the cast votes.

Individual verifiability: each voter can verify that his vote is included in the result.

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 his 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 un-

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. See **??** for a discussion on existing identity systems and their limits.

2 For every identity there exists at most one valid digital certificate.

linkability (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).

3 Related work

The seemingly most common method for counting crowds at protests is *Jacobs's method* [1, 2, 29, 37, 42]. 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. Clearly, this method is prone to errors as it is based on estimates. It provides universal verifiability since anyone can redo the counting using the same pictures. However, it is difficult to achieve individual verifiability using this method since it is hard to verify that oneself is indeed in any of the photos. We can argue that it also achieves eligibility if all the photos are taken at the same point in time, since then no one can be counted twice. Unfortunately, we can only get the peak participation with this method. When it comes to spatial and temporal eligibility, we have to rely on pictures (e.g., by determining from the pictures themselves when and where they were taken). However, this is prone to cheating — and subject to a variety of manipulation techniques — and the best we can do is to employ forensic methods to try to detect any modifications or other attempts of fraud. From a privacy perspective, all techniques based on photo or video material are necessarily not privacy-preserving unless the protesters take actions to protect themselves. For instance, there is a risk that participants are recognizable from the material if they do not wear masks.

Among more recent methods, there is an application called CrowdSize [13] that can estimate the size of a crowd in a selected area. Once the user has specified the zone, they select one of three pre-set density estimates: light, medium or dense. This is similar to the method usually used by police (e.g., during the protests in Seoul): "[p]olice presume[d] that, when sitting, six people would fill a space of 3.3 square meters [...] The same area would hold nine or 10 people when standing" [42]. This method is not verifiable unless the user takes a picture of the crowd, in which case it inherits the verifiability properties of the previous method.

In the computer vision community, there is a body of work on estimating the number of persons in a picture (e.g., the work of [46]). This class of methods requires pictures or video surveillance of the protest location during the entire protest and are thus highly privacy-invasive. They are generally based on machine learning techniques, and thus also require a training dataset to work. In the work by [46], they actually train and evaluate their algorithm on different scenes, which might make this method easier to use for protesting. Universal verifiability can be provided with this method, since someone can always recount the participants using the recorded video material. Some degree of individual verifiability could also be provided, if Alice can recognize her own face in the video, but this might still be difficult. While automatic face recognition can help verifiability, it reduces privacy. In addition, it is difficult (if not impossible) to avoid counting some people twice, thus we cannot argue for eligibility. Furthermore, it is also difficult to capture the entire location on video surveillance, and the spatial and temporal verifiability properties are reduced to those of the video.

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" [42]. In this situation it is very difficult to faithfully capture the situation. Indeed, even if the scene is taken from different angles, there is the additional problem of not counting people twice.

During the protests in Seoul [42], one company tried to estimate the number of participants using their physical analytics technology. In a nutshell, this technology scans the MAC addresses emitted from the participants' smartphones during Wi-Fi probe requests. However, in order to work this method makes many additional assumptions:

> The company presumed that about half of smartphone users usually leave on their Wi-Fi feature on and the other half switch it off, based on a separate survey on smartphone usage. It also assumed that about 20% of the smartphone signals were repetition from the same device. [42]

This method cannot provide any verifiability, as we must trust the company on performing the measurements correctly and to be honest about when and where they conducted them. In addition, as MAC address randomization is now getting a wider adoption precisely to fight against physical tracking, this method will not work any more, although some tracking of smartphones could still be possible with a different method [43].

A better proposal would be to use *IMSI catchers* (or the real cell towers of the mobile network) to count unique phones at a location. However, there are several problems with this approach. First, it will be difficult to register only the participants' phones, as many bystanders will also be counted. Thus, it will not be possible to distinguish clearly between the protesters and the counter-protesters. Second, since the phone has a unique identifier, participants might be uncomfortable to be registered in association with the event and might thus turn the device off, even though, at least with 5G, the IMSI number will not be transmitted anymore in plaintext. Finally, there will be limited verifiability, as the data recorder must be trusted to record all data in relation to the protest.

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.

In general for all of the above methods, the more a crowd spreads out, the more difficult it is to determine its size. In particular, one of the challenges is determining whether people near the event's perimeter are participants or simply bystanders [36].

One of the works most closely related to our goal is CrowdCount [12], which 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. This method has the benefit of counting everyone who has declared their presence, not just the count at the snapshot of the pictures. However, there is no verification as the service must be trusted to behave honestly, and even then, nothing prevents Bob from submitting more than once (violating eligibility). Another downside is that the service also requires an Internet connection during the event to register as a participant. This makes it vulnerable to a denial-of-service attack, e.g., Grace can shut down the cellular network or Internet backbone as a means to censor the protest.

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.

4 Definitions

4.1 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" [42]. 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 [45], 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" [41] 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" [31]. 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 [42] 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 (the 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; and 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 denote by $\mathcal{S}$ the set of all proof shares, and we will use the following notation to filter out a subset of $\mathcal{S}$ with specific cids and pids:

$$\mathcal{S}_{cid_0, pid_0} = \{(cid, t, l, pid, wid) \in \mathcal{S} \mid cid = cid_0 \wedge pid = pid_0\}.$$

Definition 3 (Participation proof). The *participation proof* of a protester with pseudonym pid who participates in a protest $\mathcal{P}$ with cause identifier cid is the set

$$\pi_{pid,\mathcal{P}} = \left\{ s \in \mathcal{S}_{cid,pid} \mid \exists p \in \mathcal{P} \colon s \sqsubseteq p \right\},$$

of all proof shares with the same protester and protest identifiers, and valid for any subprotest p of $\mathcal{P}$. We denote by Π the set of all proofs.

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. 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.

4.2 Verifiability requirements

We now try to make the properties from Section 2.2 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 match the submitted participation proofs.

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

4.3 Privacy requirements

For privacy, any temporary identifiers, such as pseudonyms, should only be reused when strictly necessary (by the principle of data minimization) for the fulfillment of verifiability properties (in Section 4.2) to prevent inference on long-term identities by repeated samples and side information. We can thus summarize the desired privacy properties as follows:

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.

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 pseudonym must be used as little as possible to still ensure the verifiability properties defined in the previous subsection (requirements P2 and P3).

4.4 Adversary model

There are three players: the protester (with identity) P, a witness (with identity) W and the time-stamping storage S (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 records when the communication occurred $t_{P,W}$. The protester P communicates with S, in which S only learns some function of the protocol data exchanged with the witness, $f(d_{P,W}(cid, P))$ for some function f, and the time of the communication ($t_{P,\mathsf{S}}$). This is illustrated in Fig. 1.

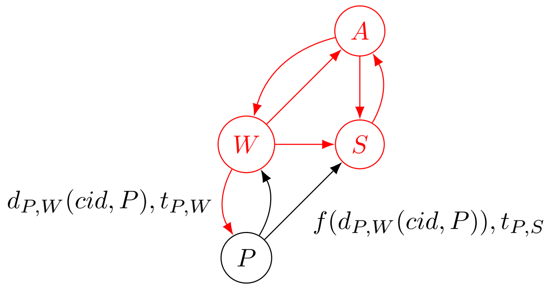

Fig. 1. An overview of the adversary model. The protester (with real identity) P and witness (with real identity) W communicate. They exchange protocol data as a function d of the protest and protester, $d_{P,W}(cid, P)$, and record the time it happened, $t_{P,W}$. The protester submits $f(d_{P,W}(cid,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

The adversary maliciously controls W. The adversary honest-but-curiously controls S, but can submit to S as everyone else. The adversary only learns the protocol data — i.e., what is sent over the channel, no auxiliary data[3].

This adversary has no access to data from outside the system, e.g., inferences that can be made from the communication layer (e.g., IP-addresses mapped to physical identities or face recognition from the protest), which means that it has only the protocol data at its disposal.

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

3 By auxiliary data we mean any data outside of the protocol, i.e., meta-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.

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

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

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 Anonymous credentials

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 he knows 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 [39], see Fig. 4.

AC must also contain a (blindable) signature scheme with the associated protocols enabling one to get a signature on a committed value ($\langle$AC.GetSig; AC.IssueSig$\rangle$) and to prove knowledge of a signature on a committed value ($\langle$AC.ProveSig; AC.VerifySig$\rangle$). This can be instantiated using CL-signatures [8], see Figs. 5 and 6.

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 $\langle$AC.ProveSig; AC.VerifySig$\rangle$ to prove the following:

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

Finally, we need a pseudo-random function, AC.PRF, such that there exists a protocol $\langle$AC.ProvePRF; 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. 7.

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 his 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 his 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 accomplice

succeeds (with non-negligible probability) he will learn the prover's secret key[4].

- 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.)

There are two lines of attempts at formalizing the above properties: one by Boureanu, Mitrokotsa, and Vaudenay [4] and another by Dürholz et al. [23].

The majority of the existing DB protocols are symmetric and thus require an honest verifier. Indeed, in this context it does not make sense to protect against the verifier as he can easily impersonate the prover as he has a knowledge of his secret key. There has been less work done in the domain of asymmetric (or public-key) DB protocols. 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 Section 6 shows, we require a DB zero-knowledge proof of knowledge, or simply proof of proximity of knowledge (PPK) [44], for discrete logarithms. For this paper, we assume the existence of such a protocol. Candidates for such a protocol include modifications of [44] or [27] to work for discrete logarithm and to provide general attribute-based anonymous credentials (not restricted to identity-based ones), respectively.

5.4 Time-stamping and storage: ledger

We need a robust time-stamping service, S, which implements the $\mathsf{S.Get}$, $\mathsf{S.Stamp}$ and $\mathsf{S.Time}$ requests such that

- $\rho \leftarrow \mathsf{S.Get}$ yields a value ρ at time t, ρ is difficult to guess before time t and $\mathsf{S.Time}(\rho) = t$;
- $\pi \leftarrow \mathsf{S.Stamp}(x)$ yields a value π at time t such that $\mathsf{S.Verify}(x, \pi) \to \top$ and $\mathsf{S.Time}(\pi) = t$.
- $\mathsf{S.Store}(x)$ stores x permanently and publicly readable.

With these building blocks, we can ensure that a message m is created within the time interval $[t_0, t_1]$. After time t_0, a user requests $\rho_{t_0} \leftarrow \mathsf{S.Get}$. Before time t_1, a user submits $h \leftarrow \mathsf{H}(m, \rho_{t_0})$ to the time-stamping service to get $\pi_{t_1} \leftarrow \mathsf{S.Stamp}(h)$.

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 $\mathsf{S.Verify}(h', \pi_{t_1}) = \top$ and $\mathsf{S.Time}(\rho_{t_0}) \geq t_0 \land \mathsf{S.Time}(\pi_{t_1}) \leq t_1$.

The value output by $\mathsf{S.Get}$ 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 $\mathsf{S.Get}$. (This can always be scaled, if $\mathsf{S.Get}$ progresses at too high pace, one can resort to only using every nth output.)

S can be instantiated by an open-membership distributed ledger (e.g., a blockchain) such as Bitcoin [38], secured via Proof-of-Work consensus, or OmniLedger [33], secured via Byzantine consensus. If a blockchain is used for S, the $\mathsf{S.Stamp}(x)$ request includes x in the blockchain and returns the identifier of the block into which x was included. The $\mathsf{S.Get}$ 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 blockchain with a high number of participants and preferably sharing the blockchain with other services. The resilience of the Byzantine consensus relies on honest participants outnumbering malicious participants. A similar reasoning can be made with Proof-of-Work consensus in which the computing power of the honest participants must be greater than the one of malicious participants. In both cases, assuming a majority of honest members in the population, the more participants the merrier. 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 S that are already provided by blockchains. First, S must be continuously extended, such as in Bitcoin in which blocks are created every 10 minutes on average. Second, S must provide *immutability* and availability to any data

4 This means that even things like functional encryption will not help.

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.

committed through $\mathsf{S.Stamp}$ to ensure verifiability of the data by anyone at any time.

6 The CROCUS Protocol

We now present CROCUS, a protocol for privately verifiable crowd counts. A prerequisite for using CROCUS is a one-to-one mapping of a person's sybil-proof identity and a cryptographic key. Since we assume these properties, we present only the result of the process of getting anonymous credentials from a CA, in Section 6.1.

The core of the CROCUS protocol consists of generating participation proofs (joining a protest, participating in witnessing, submitting proof shares to a ledger, all in Section 6.2) with an overview in Fig. 3, and then counting and verification of that count (in Section 6.3).

The entities involved in our protocol are participants and (count) verifiers.

A participant is an individual who wants to participate in a given protest. A participant can assume three different roles:

(1) The *organizer* has written a manifesto (at minimum a name) for the protest and disseminated it to others.
(2) A *protester* is attending the protest and asks witnesses to vouch for their presence.
(3) A *witness* provides proofs to protesters which state that the protester was indeed participating, constructed such that the proofs are verifiable by third parties.

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

After the protest, anyone can count the number of participants (of the protest and the protocol) and verify anyone else's count given the relevant meta data (which protest, location, time, and witness parameters used in the count). We refer to anyone counting or verifying a count as the verifier; the process is the same.

6.1 Prerequisite: anonymous credentials

Before Alice can have her participation in any event counted by CROCUS, she needs to get an anonymous credential and corresponding keys. This only needs to be repeated when the credential expires or is lost, in analogy to a passport in terms of expected intervals. 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 [34] for getting anonymous credentials, adapting only the notation to fit ours.

Setup: $(spk, ssk) \leftarrow \mathsf{Setup}$. During the setup phase, the CA creates all the needed keys. The CA generates a service public-private key-pair $(spk, ssk) \leftarrow \mathsf{AC.Setup}$ (see Fig. 5).

Registration: $sk \leftarrow \langle \mathsf{Reg}_P(spk); \mathsf{Reg}_{CA}(ssk)\rangle$. 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. 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 $\sigma \leftarrow \langle \mathsf{AC.GetSig}(spk, k, r); \mathsf{AC.IssueSig}(spk, ssk)\rangle$ (see Fig. 6). Upon success, the participant sets $sk = (\sigma, k, r)$.

6.2 Participation

The goal of CROCUS 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. The protocol phases are given in Fig. 3 and described below.

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. The organizer will then distribute 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: $(pid, t_s) \leftarrow \mathsf{Join}_P(manifesto)$. A protester who wants to join the protest will use the manifesto to compute an identifier for the cause by hashing the manifesto, $cid \leftarrow \mathsf{H}(manifesto)$ (and comparing the result to that received from the organizer, we omit this in the protocol for readibility). Afterwards, this identifier is used to create the protest-specific identifier for the protester, $pid \leftarrow \mathsf{AC.PRF}_{sk_P}(cid)$ (see Fig. 2 and Fig. 7). The protester should also receive a time-correlated random value from the time-stamping service, $t_s \leftarrow \mathsf{S.Get}$.

Joining as a witness: $t'_s \leftarrow \mathsf{Join}_W$. The witness should simply get a time-correlated random value from

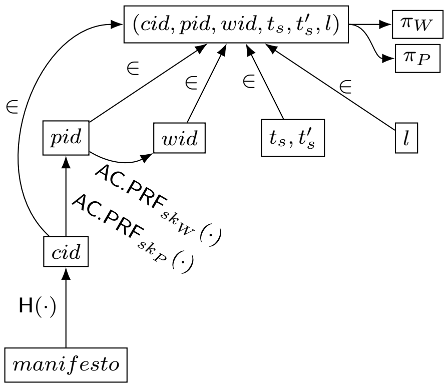

Fig. 2. Structure of a proof share. The protest (cause) identifier cid is the hash value of the manifesto. The protester P's identifier pid is computed using the protester's key sk_P and cid. The witness W's protester-specific identifier wid is computed using the witness's key sk_W and the protester's pid. t_s, t'_s are the 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.

the time-stamping service, $t'_s \leftarrow \mathsf{S.Get}$. 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: $\pi \leftarrow \langle \mathsf{Prticip}(cid, sk_P); \mathsf{Witness}(sk_W, spk)\rangle$, 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

$$\langle \mathsf{AC.ProveSig}(spk, k, r, \sigma); \mathsf{AC.VerifySig}(spk, ssk)\rangle$$

(see Fig. 6). Note that the proof of knowledge (PK) in Fig. 6 must be run as a a proof of proximity of knowledge (PPK) [44], which we do by distance bounding. 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.

Submission: $s_P \leftarrow \mathsf{Submit}_P(cid, pid, wid, t_s, t'_s, l)$. The protester commits the proof-share data to the time-stamping service and receives the proof of commitment, $t_e \leftarrow \mathsf{S.Stamp}(\mathsf{H}(cid, pid, wid, t_s, t'_s, l))$. The sooner this is done, the higher the precision for the time-dependent eligibility criterion will be for later counting. The remaining operations are not time critical.

The protester computes a non-interactive zero-knowledge (NIZK) proof ψ_{pid}, which shows the correctness of pid. More specifically,

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

Finally, the protester uploads the tuple

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

for permanent storage.

Submission: $s_W \leftarrow \mathsf{Submit}_W(cid, pid, wid, t_s, t'_s, l)$. The witness should, just as the protester, commit the proof-share data to the time-stamping service (i.e., the ledger), $t'_e \leftarrow \mathsf{S.Stamp}(\mathsf{H}(cid, pid, wid, t_s, t'_s, l))$. Then, without any time requirements, the witness computes a NIZK proof ψ_{wid} as follows:

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

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.

6.3 Count and Verification

To count or verify the participation count for a protest $\mathcal{P}$ with identifier cid_0, a verifier must download from the storage system S the set $\mathcal{S}_{cid_0}$ of all s_P and s_W tuples containing cid_0. Then from $\mathcal{S}_{cid_0}$, a verifier can build, in succession, (1) the valid proof shares $s_j^{(i)}$ for all matching pairs (s_P, s_W) corresponding to a witness i and a protester j, (2) the participation proof π_j for each protester j, (3) the set $\Pi_{\mathcal{P}}^{\varsigma,\theta}$ of eligible participation proofs for all protesters in $\mathcal{P}$, and finally, (4) the participation count, i.e., the cardinality of $\Pi_{\mathcal{P}}^{\varsigma,\theta}$.

More precisely, given

$$\mathcal{S}_{cid_0} = \{(cid, pid, wid, l, t_s, t'_s, t_c, \psi) \in \mathcal{S} \mid cid = cid_0\}$$

and a matching pair $(s_P, s_W) \in {\mathcal{S}_{cid_0}}^2$ for a witness i and a protester j with

$$\begin{aligned} s_P &= (cid_0, pid_j, wid_i, l, t_s, t'_s, t_c, \psi_i) \qquad \text{and}\\ s_W &= (cid_0, pid_j, wid_i, l, t_s, t'_s, t'_c, \psi_j),\end{aligned}$$

$$
\begin{aligned}
O \to \text{all}&\colon \text{manifesto}\\
P&\colon t_s \leftarrow \mathsf{S.Get}\\
&\quad cid \leftarrow \mathsf{H}(\text{manifesto}),\\
&\quad pid \leftarrow \mathsf{AC.PRF}_{sk_P}(cid)\\
W&\colon t'_s \leftarrow \mathsf{S.Get}
\end{aligned}
$$

Join

Participation

$$
\begin{aligned}
P \to W&\colon pid\\
P \leftrightarrow W&\colon \ \mathrm{PPK}\{(sk_P):\\
&\qquad pid = \mathsf{AC.PRF}_{sk_P}(cid),\\
&\qquad \sigma'_P = \mathsf{AC.BlindSig}(\mathsf{AC.Sign}_{ssk}(sk_P))\big\}\\
W&\colon wid \leftarrow \mathsf{AC.PRF}_{sk_W}(pid)\\
W \to P&\colon (wid, t'_s, l)
\end{aligned}
$$

(a) Join and participation.

$$
\begin{aligned}
P&\colon t_e \leftarrow \mathsf{S.Stamp}\Big(\mathsf{H}\big(pid, wid, t_s, t'_s, l\big)\Big)\\
W&\colon t'_e \leftarrow \mathsf{S.Stamp}\Big(\mathsf{H}\big(pid, wid, t_s, t'_s, l\big)\Big)\\
W&\colon \mathsf{S.Store}\Big((cid, pid, wid, t_s, t'_s, t_e, l, \pi_{wid})\Big), \quad \text{where}\\
&\quad \pi_{wid} = \mathrm{SPK}\{(sk_W):\\
&\qquad wid = \mathsf{AC.PRF}_{sk_W}(pid),\\
&\qquad \sigma'_W = \mathsf{AC.BlindSig}(\mathsf{AC.Sign}_{ssk}(sk_W))\big\}\\
&\qquad\quad (cid, pid, wid, t_s, t'_s, l)\\
P&\colon \mathsf{S.Store}\Big((cid, pid, wid, t_s, t'_s, t_e, l, \pi_{pid})\Big), \quad \text{where}\\
&\quad \pi_{pid} = \mathrm{SPK}\{(sk_P):\\
&\qquad pid = \mathsf{AC.PRF}_{sk_P}(cid),\\
&\qquad \sigma'_P = \mathsf{AC.BlindSig}(\mathsf{AC.Sign}_{ssk}(sk_P))\big\}\\
&\qquad\quad (cid, pid, wid, t_s, t'_s, l)
\end{aligned}
$$

(b) Submission.

Fig. 3. An overview of CROCUS participation. The organizer O broadcasts the manifesto. The protester P, witness W and their computations are as in Fig. 2. Finally, both P and W submit the proof shares to a public ledger for permanent storage S. Note that pidalways refers to the protester whose presence is being witnessed.

the verifier can build a valid proof share $s_j^{(i)}$ certified by i for j as follows: verify ψ_i and ψ_j, let

$$
\begin{aligned}
t &= [\max(t_s, t'_s), \min(t_c, t'_c)] \qquad \text{and}\\
s_j^{(i)} &= (cid_0, pid_j, wid_i, l, t),
\end{aligned}
$$

as in Definition 2, check that $s_j^{(i)}$ is valid (i.e., happened during and at the location of the protest), as in Definition 2.

Then the set of all valid proof shares for a protester j constitutes its participation proof π_j, as in Definition 3, and the verifier thus can derive the set of (ς, θ)-eligible participation proofs $\Pi_{\mathcal{P}}^{\varsigma,\theta}$ for all protesters for the protest $\mathcal{P}$, as in Definition 4. Finally, the participation count $|\Pi_{\mathcal{P}}^{\varsigma,\theta}|$ is the cardinality of this set by the same Definition 4.

In the case of trusted witnesses, each such trusted witness must publish or otherwise inform the verifier of which proof shares they have signed, e.g., by giving a list of all such proof shares or digitally signing each proof share[6].

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 and as long as these 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.

7 Security and privacy analysis

7.1 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

6 To achieve witness privacy in this situation, one could employ a group or ring signature scheme for a set of trusted witnesses, e.g., members of an independent journalist association. Then one learns that at least one member of this set of trusted witnesses must have been there.

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 7.2 that the adversary cannot drop submitted proof shares and thus cannot decrease the count. As indicated in the adversary model (Section 4.4), 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 in this section, as she naturally has an incentive to increase the count, as the organizer and a participant.

7.1.1 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 $\mathsf{S.Get}$ 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. 2). The unpredictability of $\mathsf{S.Get}$ 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 $\mathsf{S.Time}$.

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 ($\mathsf{S.Stamp}$), 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.

7.1.2 Linkability

Requirement V1.3 prevents 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 $\mathsf{AC.PRF}$, Alice must produce a new key sk'_P such that the verifier[7] accepts the proof

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

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.

7.1.3 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 trust in the witness's honesty. 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 a DF attack against the DB protocol. We assumed DF resistance for the DB protocol, so Alice cannot succeed with more than negligible probability.

Forging a witness's signature on a proof is equivalent to breaking linkability above (Section 7.1.2). As above, this is reduced to the security of the AC scheme.

Finally, 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, (by assumption) 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.

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

7.1.4 Designated use

Requirement V1.4 is 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 protest, with a different manifesto, one must find a second pre-image $manifesto'$ such that $cid = \mathsf{H}(manifesto) = \mathsf{H}\big(manifesto'\big)$.

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

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

and

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

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

7.2 Individual and universal verifiability

Requirement V3 requires 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 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 and storage (S, Section 5.4).

We assumed an honest-but-curious adversary controlling S[8]. 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. As for individual verifiability, the security of universal verifiability is reduced to the properties of the ledger and storage; but universal verifiability also depends on the eligibility verification.

8 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.

7.3 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]).

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 $pid' = \mathsf{AC.PRF}_{sk}\big(cid'\big)$ (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 $cid, pid, wid, t_s, t'_s, l, \psi_{pid}, \psi_{wid}$. We assume that the *cid* will be used by many individuals, as it is the cause identifier. And thus it cannot be used to uniquely identity any individual.

We assume that 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 is interesting. Thus l is not uniquely identifying any individual protester or witness. Likewise, thanks to the constraints in the time-stamp granularity (Section 5.4), t is not uniquely identifying either.

8 Performance

8.1 Smartphones and smartcards

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. 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 [30] have shown that Android devices are now fast enough to efficiently implement Privacy Enhancing Technologies (PETs), with a Samsung Galaxy S i9000 being 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 [3] 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 actually very similar to Anon-Pass [34], 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 the authors was using 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 milliseconds while it is 23.4 milliseconds 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 milliseconds while the time needed for the witness would be only 7.9 milliseconds.

8.2 Distance-bounding anonymous credentials

The distance-bounding chips[9] currently available on the market can already enable proofs of proximity for any range of up to 200 meters. One of the objectives is to be able to integrate them in phones or smartcards in the near future, and even phones with off-the-shelf hardware running in RFID-emulation mode have shown promising results. For instance in [28], the authors have demonstrated that it is already possible to implement distance-bounding protocols as a standard Android application on existing smartphones. More precisely, they have proposed three different implementations of the Swiss-Knife protocol using 32 challenge-response rounds ranging from a basic implementation running a the application layer to an advanced one running in RFID-emulation mode. The results obtained already show that a relay attack can be detected if the adversary introduces a delay of more than 1.5 ms when performing such an attack (most of the existing attacks introduce delays of at least tens of milliseconds).

8.3 Ledger (blockchain) efficiency

As an example, consider a protest with 1 000 000 participants. If we use 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 [33], which can do approximately 1500 transactions per second, it takes at least 11 minutes to process all the proof shares.

If we do not use trusted witnesses, but instead a threshold $\theta = 1000$, then it will take at least 7 days before all transactions are committed to the blockchain. While this already takes longer than counting votes in national elections, the threshold is still very low. For large groups organizing online to collectively pretend to participate at a physical protest, the threshold would have to be much higher. Considering that online forums can facilitate mass coordination, the threshold approach with only untrusted witnesses seems not only inefficient but also potentially ineffective for verifiability.

9 https://www.3db-access.com

9 Discussion: limitations of assumptions

9.1 Equipment

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[10].

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 a 1) mapping between an identity and a physical person (in the form of an anonymous identity credential, functioning as a pseudonym for privacy properties) and 2) only one such credential per identity (token of 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 [40] could ensure 2). Again, this is not yet realized, but with the current pace for 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[11] 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.

9.2 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) [32]. 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 requiring the participant and witness to hold their phones together for each proof share.

9.3 Adversaries

Our adversary model considers only protocol data, no auxiliary data. Against this adversary our scheme is se-

10 Official statistics, in Swedish: https://www.bankid.com/assets/bankid/stats/2018/statistik-2018-04.pdf.
11 http://ec.europa.eu/eurostat/web/quality/european-statistics-code-of-practice

cure. 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 the canonical CROCUS with trusted witnesses, as in any trusted third party, it simply reduces to their trustworthiness.

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 location 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.

10 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.

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

[2] 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).

[3] 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).

[4] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. "Practical and provably secure distance-bounding." In: *Journal of Computer Security* 23.2 (2015), pp. 229–257.

[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] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. "A formal approach to distance-bounding RFID protocols." In: *International Conference on Information Security*. Springer. 2011, pp. 47–62.

[24] 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).

[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] Sébastien Gambs, Carlos Eduardo Rosar Kos Lassance, and Cristina Onete. "The Not-so-distant Future: Distance-Bounding Protocols on Smartphones." In: *14th Smart Card Research and Advanced Application Conference*. Bochum, Germany, Nov. 2015.

[29] 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).

[30] 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.

[31] 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).

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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).

[37] 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).

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

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

[40] 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.

[41] 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).

[42] 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).

[43] 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).

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

[45] 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).

[46] 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.

[47] 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.

.1 Distance bounding and location proofs

Some location-based services (LBSs) 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. In most of the existing schemes, the location of a user (device) is determined by the device itself (e.g., through GPS) and forwarded to the LBS provider. One of the main drawbacks of this approach is that a user can cheat by having her device transmit a false location. Therefore, it is possible for a user to be inappropriately granted access to a particular resource while being thousands of kilometers away.

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 [35, 47]. 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, although their objective is quite different as it aims at verifying a global property of the population (i.e., crowd estimation) in contrast to checking the location claim made by a user, which is an individual property.

Another difference is that CROCUS operates in a more adverse environment. CROCUS must provide *universal verifiability*, this means that all proofs must be available to and verifiable by anyone. One problem here is that we have multiple verifiers who might not trust the same witnesses. The incentives to cheat are also bigger and consequently the thresholds for collusion are much higher.

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. 4 to 8.

```
function P.Commit($x, r$)
    return $g^x h^r$
```

Fig. 4. Pedersen's commitment scheme [39]. 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 $\mathbb{Z}_q$.

```
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 $\perp$                                   ▷ $A$ malformed
    else if $e(a, Y) \neq e(g, b) \vee e(A, Y) \neq e(g, B)$ then
        return $\perp$                                   ▷ $b$ or $B$ malformed
    else if $e(X, a) \cdot e(X, b)^m \cdot e(X, B)^r \neq e(g, c)$ then
        return $\perp$                                   ▷ $c$ malformed
    return $\top$
```

Fig. 5. The CL-signature scheme [8]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \rightarrow G_T$ 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($pk, x, y, \pi$)
    if $e(g^x \cdot pk, \pi) \neq e(g, g)$ then
        return $\perp$
    else if $y \neq e(g, \pi)$ then
        return $\perp$
    return $\top$
```

Fig. 7. Verifiable random function [21]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \rightarrow G_T$ be a bilinear map.

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

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

Fig. 6. Protocols for CL anonymous credentials [8]. Let $G = \langle g \rangle, G_T = \langle g_T \rangle$ be groups of prime order q. Let $e \colon G \rightarrow G_T$ be a bilinear map.

CHKL.ProvePRF(k, x)		**CHKL.VerifyPRF**(y)
$y \leftarrow$ **DY.PRF**(k, x)		
	$\mathbf{PK}\{(k) : y =$ **DY.PRF**$(k, x)\}$	

Fig. 8. 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 $e \colon G \rightarrow G_T$ be a bilinear map.