ANONIZE: A Large-Scale Anonymous Survey System

Susan Hohenberger
Johns Hopkins University
susan@cs.jhu.edu

Steven Myers
Indiana University
samyers@indiana.edu

Rafael Pass
Cornell University
rafael@cs.cornell.edu

abhi shelat
University of Virginia
abhi@virginia.edu

***Abstract*—A secure *ad-hoc survey* scheme enables a survey authority to independently (without any interaction) select an ad-hoc group of registered users based only on their identities (e.g., their email addresses), and create a survey where only selected users can *anonymously* submit *exactly one* response. We present a formalization of secure ad-hoc surveys and a provably-secure implementation in the random oracle model, called ANONIZE. Our performance analysis shows that ANONIZE enables securely implementing million-person anonymous surveys using a single modern workstation. As far as we know, ANONIZE constitutes the first implementation of a large-scale secure computation protocol (of non-trivial functionalities) that scales to millions of users.**

I. Introduction

We study the basic conflict between anonymity and authenticity in large network settings. Companies, universities, health providers and government agencies routinely conduct asynchronous and real-time data collection surveys for targeted groups of users over the Internet. To do so, they aim for *authenticity* (i.e., ensuring that only the legitimate users can participate in the data collections) and *anonymity* (i.e., ensuring that the there is no link between the legitimate user and his/her data so that users are more likely to submit honest feedback). The intrinsic conflict between these two goals may result in users self-censoring or purposely biasing data they submit.

A simple example is a course evaluation for a university class. A typical implementation of such a survey requires a trusted third party (such as the university, or some external party) to ensure that feedback is collected anonymously from the participants and that only authorized participants, i.e., the students enrolled in a particular class, can submit feedback for that class. In such trusted-party implementations, students are required to authenticate themselves with their university IDs and thus leave a link between their evaluation and their identity; they are trusting the survey collector to keep such links private.

Assuming that the survey collector acts as a trusted third party is dangerous. Even if the survey collector intends to keep the links between users and their surveys private, its computer may be stolen or broken into, and the information leaked. For instance, in 2009, a computer at Cornell was stolen, containing sensitive personal information, such as name and social security number, for over 45,000 current and former university members [1]. Additionally, even if users have full confidence in the the trusted third party, and in particular, its ability to keep its data *secure*, developing an anonymous survey system using such a trusted party still requires some care. For example, in the implementation of course reviews at the University of Virginia, side channel information indicating who has already filled out the survey may leak information about the order in which students participate. Later, the order of the students' comments in the aggregated responses may be correlated to break anonymity [2].

Furthermore, in many situations, jurisdictional boundaries or legal requirements make it unfeasible to rely on solutions with external trusted third parties: it may be illegal to store sensitive patient information on a third-party system; similarly, many countries do not permit sensitive data to be stored on servers run by foreign corporations due to the potential for this data to be seized [3].

For these reasons, we seek cryptographic solutions to the problem of anonymous surveys that offer security guarantees where anonymity and authenticity hold *without needing to trust a third party*.

Cryptographic voting techniques described in prior work may offer a partial solution to this problem (see e.g., [4], [5], [6], [7], [8]). In such schemes, each survey consists of two steps: 1) users authenticate themselves to a server and anonymously check out a *single-use* "token"; the token itself carries no link to the user's identity. 2) a user can then use her token to participate in the specified survey. Such schemes provide good anonymity *assuming that* users actually separate steps 1 and 2 with a reasonably long time lag (otherwise there is a clear time link between the user and its data). But if users are required to separate the two steps by, say, a day, the ease-of-use of the survey is significantly hampered and become much less convenient than "non-anonymous" surveys (or anonymous surveys employing a trusted third party). Additionally, the extra steps required to authenticate for each survey may be onerous. Consequently, such techniques have gained little traction.

A. Our innovation: electronic ad-hoc surveys

In this paper, we consider a general solution to the problem of anonymously collecting feedback from an authenticated group of individuals by introducing the notion of an *ad-hoc survey*. The "ad-hoc" aspect of this notion means that *anyone* can select a group of individuals and create a survey in which those and only those individuals can complete the survey *at most once*; additionally, the survey initiator can initiate this survey knowing only the identities (e.g., the email addresses) of the users in the ad-hoc group—no further interaction between the survey initiator and the users is required.[1] As such, our method provides essentially the same ease-of-use as traditional (non-anonymous) electronic surveys (and it thus is expected to increase user participation and make the feedback submitted more valuable).

As we demonstrate, ad-hoc surveys admit practical and efficient solutions for very large surveys: we present an ad-hoc survey scheme, ANONIZE, a proof of security for the cryptographic protocols in ANONIZE, and an implementation of the protocol. ANONIZE supports millions of "write-in" (i.e., collection of arbitrary strings of data) surveys in minutes. As far as we know, this is the first implementation of a provably-secure[2] multi-party protocol that scales to handle millions of users. Additionally, we prove security of our scheme even if the adversary participates in an arbitrary number of concurrent surveys.

B. Ad-hoc Surveys in more detail

In more details, there are three parties in an ad-hoc survey system: a registration authority (RA) that issues master user tokens, a survey authority (SA) that can create surveys, and users that provide survey data. A user must first register with the RA and retrieve a secret "master user token". This is a single token that can be used for all future surveys the user participates in. Anyone can act as an SA by choosing a uniquel survey ID and publishing a list of identities that are permitted to participate in that survey. The list of identities that can participate in a particular survey can grow dynamically, and the SA can create a survey without any interaction with others. Finally, a user who is on the list of valid identities for a survey can *non-interactively* submit a response to the survey by simply routing one message to the SA (through an anonymous network like Tor, or anonymous proxy relay).

To exemplify this approach and informally discuss the anonymity/authenticity properties it provides, we consider the course evaluation scenario.

1) Student Registration: When a student is asked to set-up his college/university account information (while proving his identity using traditional, non-electronic, methods), the student also generates an *unlinkable master user token* that is tied to his school email identity (e.g., his email address). This step can also be done at a later stage if the student desires (or if the student loses his credential), but it only needs to be done *once*.

2) Course Survey Setup: Whenever a course administrator wishes to set-up a course survey, she generates a *survey key* based only on the actual identities (e.g., the email addresses) of the course participants.

3) Survey Execution: Upon filling out a survey with its associated survey key, the student's client (either computer or smart phone) combines the survey key and her *master user token* to generate an unlikable *one-time token* that she can use to complete the survey. The one-time token satisfies two properties: 1) it carries no link to the student's identity (thus we have anonymity), and 2) for a given survey key, the student can obtain at most one such token (and thus we ensure that a student can only complete the survey once[3]). The results of the survey can now be tabulated, and, possibly announced.

We emphasize that once Step 1 has been done (presumably once the students enroll into college), Steps 2 and 3 can be repeatedly performed. The participants do not need to check-out new single-use tokens for each survey; rather their client uses the master user token to create a unique single-use token for this survey *without any interaction* (that could deanonymize the student).

Part of our contribution is to precisely define security properties of ad-hoc surveys such as anonymity (intuitively, that there is no link between users and the surveys they submit), and authenticity (intuitively, that only authorized users can complete the survey, and they can complete it only once). As mentioned, we are interested in providing security not only for a single survey, but also if an attacker participates in many surveys, be they in the past, concurrent, or in the future. A common approach for defining security in such circumstances is to formalize the notion of secure ad-hoc surveys within the framework for Universal Composability [9]. Doing so permits one to analyze the protocol under a single instance and deduce that it also remains secure under concurrent executions. Unfortunately, there are well-known inefficiencies with this approach. Rather, to enable an efficient implementation, we provide direct game-based definitions of security and directly analyze the security of our protocol under concurrent executions—this is analogous with other cryptographic

[1] Before users can complete a survey, we additionally require them to register their identity. We emphasize that this registration is done only once and can be used for any number of subsequent surveys.

[2] By "provably-secure", we only refer to the cryptographic protocol.

[3] Our systems support the (optional) ability for the user to change her response (before the voting deadline) in a manner that replaces her previous submission, but in no other way leaks any information about her identity.

game-based definitions, e.g., blind signatures [10]; we emphasize that although related notions of anonymity and authenticity have been defined in the literature for other applications, our setting is considerably more complex and thus the actual definitions are different.

C. Anonize in more detail

Our system is constructed in two steps. We first provide an *abstract* implementation of secure ad-hoc surveys from generic primitives, such as commitment schemes, signatures schemes, pseudo-random functions (PRF) and generic non-interactive zero-knowledge (NIZK) arguments for NP[4]. We prove the security of the abstract scheme based on the assumption that all generic primitives employed are secure. Note that we have taken explicit care to show that our schemes remain secure even when the adversary initiates *many concurrently* executing sessions with the system.

In a second step we show that (somewhat surprisingly) the generic scheme can be instantiated with *specific* commitment schemes, signatures schemes, PRF and NIZKs to obtain our efficient secure ad-hoc survey scheme ANONIZE (which now is based on specific computational assumptions related to the security of the underlying primitives in the Random Oracle Model). The surprising aspect of this second step is that our generic protocol does *not* rely on the underlying primitives in a black-box way; rather, the NIZK is used to prove complex statements which require code of the actual commitments, signatures and PRFs used. In this second step, we rely on ideas similar to those underlying efficient constructions of anonymous credentials in bilinear groups [11], [12], although our constructions differ in a few ways. As far as we know, our scheme is also one of the first implementations of a cryptographic scheme that is concurrently-secure.

Let us briefly provide a high-level overview which omits several important features, but conveys the intuition of our abstract protocol (we assume basic familiarity with the concepts of commitment schemes, signature schemes, PRFs and NIZKs).

1) Registration: A user with identity id registers with the RA by sending a commitment to a random seed s_{id} of a pseudo-random function (PRF) F and providing a NIZK that the commitment is well-formed. If the user has not previously been registered, the RA signs the user's name along with the commitment. The signature returned to the user is its "master user token". The security property required here is weaker than that of a blind signature.

2) Survey: To create a survey, an SA publishes a list of signed user identities along with a survey id, v_{id}.

3) Response: To complete a survey for survey id v_{id}, a user id generates a *single-use token* $F_{s_{id}}(v_{id})$ (by evaluating the PRF on the seed s_{id} with input v_{id}) and presents a NIZK that it "knows a signature by the RA on its identity id and a commitment to a seed s_{id}" and that it "knows a signature by the SA on its id" and that the single-use token is computed as $F_{s_{id}}(v_{id})$. The user's actual survey data will be part of and thereby authenticated by this NIZK.

Roughly speaking, the NIZK proof in the survey completion step ensures that only authorized users can complete the survey, and that they can compute at most one single-use token, and thus complete it at most once. [5] Anonymity, on the other hand, roughly speaking follows from the fact that neither the RA nor the SA ever get to see the seed s_{id} (they only see commitments to it), the zero-knowledge property of the NIZKs, and the pseudo-randomness property of the PRF.

Proving this abstract protocol secure is non-trivial. In fact, to guarantee security under concurrent executions, we introduce and rely on a new notion of a *simulation-extractable NIZK* (related to simulation-sound NIZK [13] and simulation-extractable interactive zero-knowledge arguments [14], [15]).

To enable the second step of our construction (i.e., the instantiation of the abstract protocol using specific primitives), we demonstrate a simple and efficient way of implementing simulation-extractable NIZK in the Random Oracle Model by relying on the Fiat-Shamir Heuristic [16]. Finally, the key to the construction is choosing appropriate commitments, signatures and PRF that can be "stitched together" so that we can provide an efficient NIZK for the rather complex statement used in the abstract protocol.

D. Related notions and techniques

Ad-hoc surveys are related to, but different from, a number of primitives previously considered in the literature such as *group signatures*, *ring signatures*, *voting schemes* and *anonymous credentials*. Roughly speaking, group [17], [18], [19] and ring [20] signatures allow members of a set of users to sign messages in a way that makes it indistinguishable who in the set signed the message (in the case of group signatures the set is fixed [6], whereas in the case of ring signatures the set can be selected "ad-hoc"). This property is similar to the

[4] As we show, we actually need a new variant of standard NIZKs.

[5] If the user wants to replace her survey response before the deadline and this is allowed by the system, then she can create a new NIZK with new data for the same $F_{s_{id}}(v_{id})$ value. The old survey with this value can be deleted.

[6] The desirability of making group signatures dynamic was addressed by Bellare, Shi and Zhang [21]. Their solution, however, requires that every group member or potential group member has their own personal public key, established and certified, e.g., by a PKI, independently of any group authority. Our ad-hoc survey solution does not require this.

anonymity property of ad-hoc survey, but unfortunately, the authentication property these notions provide is insufficient for our setting—in a ring signature scheme, a user may sign multiple messages with impunity which corresponds to the ability to complete the survey multiple times in our context. Voting schemes [4], [5], [6], [7], [8] on the other hand do provide both the anonymity and the authenticity properties we require; however, they do not allow for the authenticated users to be selected ad-hoc for multiple elections.

An anonymous credential system [22], [23], [24], [11] allows users to obtain credentials from authorities and to anonymously demonstrate possession of these credentials. In essence such systems provide methods for providing, a "zero-knowledge proof of knowledge of a signature on a set of attributes." As mentioned, the NIZKs we use rely on intuitions similar to those used in constructions of anonymous credentials (most closely related to [11] and the electronic cash/token extensions in [25], [12]), but we have different goals and rely on different complexity assumptions. Moreover, since anonymous credentials typically are not analyzed under concurrent executions, we must develop new techniques for the security analysis.

E. Our Implementation

One of the key points of our system is that it can be implemented and can easily handle large numbers of users with moderate resources. The computational costs on the users are quite low as well, with a typical desktop being able to compute the worst-case scenario in under a few seconds, using a single core of the machine. Thus we argue our system scales to manage that vast majority of practical surveying needs at costs that are easily affordable.

II. Ad-hoc Surveys

An Ad-hoc Survey Scheme is a protocol involving three types of players:

— A *single* Registration Authority (RA).
— One or multiple Survey Authorities (SA).
— Users; each user is associate with a *public* user identity id (e.g., its email address).

We assume that the RA has the ability to set up a secure session (private and authenticated) with the user associated with a particular user identity. Each user additionally has the ability to setup an anonymous connection to the SA when returning their survey.

An *ad-hoc survey scheme* is a tuple of algorithms $(\mathsf{GenRA}, \mathsf{GenSA}, \mathsf{Reg}^{\mathsf{RA}}, \mathsf{Reg}^{U}, \mathsf{GenSurvey}, \mathsf{Authorized}, \mathsf{Submit}, \mathsf{Check})$ which we formalize shortly. To gain some intuition, let us first explain how these algorithms are intended to be used in a system and informally explain what types of security requirements we want from the algorithms.

System Set-up:

— The RA generates a public key-pair $\mathsf{pk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}} \leftarrow \mathsf{GenRA}(1^n)$; $\mathsf{pk}_{\mathsf{RA}}$ is made public and $\mathsf{sk}_{\mathsf{RA}}$ is secretly stored by the RA.
— Each SA generates a public key-pair $\mathsf{pk}_{\mathsf{SA}}, \mathsf{sk}_{\mathsf{SA}} \leftarrow \mathsf{GenSA}(1^n)$; $\mathsf{pk}_{\mathsf{SA}}$ is made public and $\mathsf{sk}_{\mathsf{SA}}$ is secretly stored by the SA.[7]
— For systems that require the use of a Common Reference String (CRS); a CRS is generated and made publicly available. For simplicity of notation, we omit the CRS in all the procedures below and simply assume that all these procedures get the CRS as an input. Likewise, for systems in the Random Oracle model, we assume the procedures below have access to the Random Oracle.

User Registration: To use the system, users need to register with the RA; at this point the user and the RA execute the protocol $(\mathsf{Reg}^{\mathsf{RA}}, \mathsf{Reg}^{U})$ which allows the user to check out an *unlinkable "master credential"*. A user with identity id proceeds as follows:

1) The user sets up a secure session with the RA.
2) The RA checks that user identity id previously has not been registered. If it has, the RA closes the session. Otherwise, the RA and the user invoke the interactive protocol $(\mathsf{Reg}^{\mathsf{RA}}, \mathsf{Reg}^{U})$ on the common input $1^n, id$.
3) If the protocol ends successfully, the RA stores that user identity id has been registered, and the user secretly stores the output as $cred_{id}$.

Survey Registration: Whenever an SA wants to set-up a survey with identifier sid, it generates a *"survey public-key"* based on the identities of the participants (and its own secret key). More precisely, the SA on input a survey identifier sid and a list L of user identities (they may be previously registered or not) computes and makes public $\mathsf{pk}_{\mathsf{sid}} \leftarrow \mathsf{GenSurvey}(1^n, \mathsf{sid}, L, \mathsf{sk}_{\mathsf{SA}})$.

Completing a Survey: Given a registered survey with identifier sid and its associated public-key $\mathsf{pk}_{\mathsf{sid}}$, each "authorized" user id_i can combine its master credential $cred_{id}$ with the survey identifier sid and public-key $\mathsf{pk}_{\mathsf{sid}}$ to generate an unlikable *"one-time token"* that it can then use to make a submission in the survey. Roughly speaking, the "one-time token" satisfies two properties: 1) it carries no link to the students identity (thus we have anonymity), and 2) for a given "survey key", the student can obtain at most one such token (and thus can only submit one response).

More precisely, user id with master credential $cred_{id}$ submits the message m as the completed survey by privately executing the algorithm $\mathsf{Sub} = (\mathsf{tok}, m, \mathsf{tokauth}) \leftarrow \mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, m, cred_{id})$

[7] Our security properties hold even if new SAs are added on-the-fly.

and then submitting Sub to the SA through an anonymous channel; tok is the "one-time token", and tokauth is an authenticator required to bind the message m to the one-time token, and to ensure uniqueness of the one-time token. SA checks whether the submission is correctly computed by executing $\mathsf{Check}(pk_{\mathsf{SA}}, pk_{\mathsf{RA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, \mathsf{Sub})$; if it outputs accept it stores the submission. If a submission with the same tok has been previously stored (i.e., if a Sub of the form $(\mathsf{tok}, m', \mathsf{tokauth}')$ has already been stored, the old record is removed. (Or alternatively, the new Sub is not stored.)

Announcing the results: Once all the submissions have been collected, the SA may (depending on external privacy requirements) publish a list of all stored submissions $\mathsf{Sub} = (\mathsf{tok}, m, \mathsf{tokauth})$.

Audit Procedures: The system also includes audit procedures. First, Users can check that their submission was "counted" by simply inspecting that their submission is output. Second, a User may use $\mathsf{Check}(pk_{\mathsf{SA}}, pk_{\mathsf{RA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, \mathsf{Sub})$ to check whether Sub is a valid submission (i.e., user can check that there is no "ballot/survey-stuffing"). Finally, to ensure that a survey is not targeted to a particular user (for de-anonymization purposes), the user may use function $\mathsf{Authorized}(pk_{\mathsf{SA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, \mathsf{id}')$ to check whether user id' is also authorized for survey sid with public key $\mathsf{pk}_{\mathsf{sid}}$.

Key features and Security Properties: A crucial aspect of an ad-hoc survey is the *privacy* property: even if the RA and SA are arbitrarily corrupted (and in collusion) they cannot learn anything about how particular users answered submissions (or even learn correlations between groups of users). The key *security* property of our ad-hoc survey is that only authorized users can complete a survey, and furthermore they can complete it at most *once*.

A. Definition of an Ad-hoc Survey

Definition 1: An *ad-hoc survey scheme* Γ is an 8-tuple of PPT algorithms and interactive PPTs (GenRA,GenSA,$\mathsf{Reg}^{\mathsf{RA}}$,$\mathsf{Reg}^{U}$, $\mathsf{GenSurvey}$, $\mathsf{Authorized}$, Submit, Check) where

- $\mathsf{GenRA}(1^n)$ outputs a key-pair $\mathsf{pk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}}$.
- $\mathsf{GenSA}(1^n)$ outputs a key-pair $\mathsf{pk}_{\mathsf{SA}}, \mathsf{sk}_{\mathsf{SA}}$.
- $\mathsf{Reg}^{\mathsf{RA}}(\mathsf{sk}_{\mathsf{RA}}, 1^n, \mathsf{pk}_{\mathsf{RA}}, id_i)$ is an interactive PPT that outputs either success or fail.
- $\mathsf{Reg}^{U}(1^n, \mathsf{pk}_{\mathsf{RA}}, id)$ is an interactive PPT that outputs a bitstring $cred_{id}$ or fail.
- $\mathsf{GenSurvey}(1^n, \mathsf{sid}, L, \mathsf{sk}_{\mathsf{SA}})$ outputs a bitstring $\mathsf{pk}_{\mathsf{sid}}$. Here sid is a unique arbitrary identifier and L is a description of the set of users eligible to participate in the survey.
- $\mathsf{Authorized}(pk_{\mathsf{SA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, id)$ outputs either YES or NO.
- $\mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, m, cred_{id})$ outputs $\mathsf{Sub} = (\mathsf{tok}, m, \mathsf{tokauth})$.
- $\mathsf{Check}(pk_{\mathsf{RA}}, pk_{\mathsf{SA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, \mathsf{Sub})$ outputs either accept or fail.

A remark on the Authorized procedure: We are interested in schemes where the description of the authorized procedure makes it possible to naturally interpret the *set* of users that are allowed to complete a survey (and indeed, our constructions fall into this category). For instance, the description of the Authorized procedure specifies a list of user identities, or specifies a list of user identities with wildcard (e.g., $*@*.cornell.edu$). In our specific implementation, the public key for the survey $\mathsf{pk}_{\mathsf{sid}}$ consists of a list of authorized users.

B. Correctness

We proceed to define what it means for an ad-hoc survey scheme to be *correct*. The following definition requires that for every set of users L, and every user $\mathsf{id} \in L$, if an SA sets up a survey for L, and if the user correctly registers with the RA, then the user will be authorized to complete the survey; furthermore, for every submission m, if user id correctly submits m, this submission will pass the check.

Definition 2: An ad-hoc survey scheme Γ is *correct* if there exists a negligible function $\mu(\cdot)$, such that the following experiment outputs fail with probability at most $\mu(n)$ for every $n \in \mathbb{N}$, $\mathsf{sid}, m \in \{0,1\}^n$, set L of n-bit strings, $id \in L$:

- $(\mathsf{vk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}}) \leftarrow \mathsf{GenRA}(1^n)$
- $(\mathsf{vk}_{\mathsf{SA}}, \mathsf{sk}_{\mathsf{SA}}) \leftarrow \mathsf{GenSA}(1^n)$
- Set $(out_{\mathsf{RA}}, out_{\mathsf{U}})$ to the result of the protocol $(\mathsf{Reg}^{\mathsf{RA}}(\mathsf{sk}_{\mathsf{RA}}, 1^n, \mathsf{vk}_{\mathsf{RA}}, id), \mathsf{Reg}^{U}(1^n, \mathsf{vk}_{\mathsf{RA}}, id))$
- Output fail if either out_{RA} or out_{U} is fail, otherwise let $cred_{id} = out_{\mathsf{U}}$.
- $\mathsf{vk}_{\mathsf{sid}} \leftarrow \mathsf{GenSurvey}(1^n, \mathsf{sid}, L, \mathsf{sk}_{\mathsf{SA}})$
- Output fail if $\mathsf{Authorized}(\mathsf{vk}_{\mathsf{SA}}, \mathsf{sid}, \mathsf{vk}_{\mathsf{sid}}, id) = \mathsf{NO}$.
- $\mathsf{Sub} \leftarrow \mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{vk}_{\mathsf{sid}}, m, cred_{id})$
- Output $\mathsf{Check}(\mathsf{vk}_{\mathsf{SA}}, \mathsf{vk}_{\mathsf{RA}}, \mathsf{sid}, \mathsf{vk}_{\mathsf{sid}}, \mathsf{Sub})$

C. Privacy and Security

The following definition stipulates that the SA(s) and RA and malicious users, even if arbitrarily corrupted, cannot distinguish the submissions of two authorized honest users, even for an adaptively chosen participant list, user identities and submission messages, and even if they may see the submission messages of the two participants for any messages of their choosing, in any *other* survey of its choice. Thus, even if an attacker knows what submissions correspond to which users in any surveys of its choice (before and after the survey

of interest), it still cannot identify what these users submitted for a survey of interest. The definition mirrors the definition of CCA-secure encryption: we give the attacker the opportunity to generate an arbitrary public-key for the RA, pick two user identities $\mathsf{id}_0, \mathsf{id}_1$, ask the users to register with him, and then make oracle queries to the users' Submit procedure. Finally, the attacker selects a survey consisting of a public-key for a SA, a sid and a public key for the survey $\mathsf{pk}_{\mathsf{sid}}$ such that $\mathsf{id}_0, \mathsf{id}_1$ are both authorized (for which it has not yet queried the Submit oracle on sid), a pair of messages m_0, m_1, and then sees two submissions. The attacker must guess whether the two submissions correspond to ones from id_0 and id_1 or from id_1 and id_0 respectively; the attacker continues to have oracle access to the users' Submit procedure during this decision-making phase but cannot make queries on the sid.

Definition 3: An ad-hoc survey scheme Γ is *unlinkable* if for every non-uniform PPT A the ensembles $\{\mathsf{EXEC}^0(1^n, A)\}_{n\in\mathbb{N}}$, $\{\mathsf{EXEC}^1(1^n, A)\}_{n\in\mathbb{N}}$ are computationally indistinguishable where $\mathsf{EXEC}^b(1^n, A)$ is defined as follows:

$\mathsf{EXEC}^b(1^n, A)$

- $(\mathsf{vk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}}), z \leftarrow A(1^n)$
- $A(1^n, z)$ *concurrently interacts with* $\mathsf{Reg}^U(1^n, \mathsf{vk}_{\mathsf{RA}}, \mathsf{id})$ *for any two different ids* id_0 *and* id_1. *Whenever an interaction with some* $\mathsf{Reg}^U(1^n, \mathsf{vk}_{\mathsf{RA}}, \mathsf{id}) \neq \perp$ *completes, for the remainder of the experiment,* A *gets oracle access to* $\mathsf{Submit}(1^n, \cdot, \cdot, \cdot, cred_{\mathsf{id}})$. *Next,* A *outputs a target survey:*
- $(\mathsf{vk}_{\mathsf{SA}}, \mathsf{sid}, \mathsf{vk}_{\mathsf{sid}}, \mathsf{id}_0, \mathsf{id}_1, m_0, m_1, z') \leftarrow A(1^n, z)$
- *Output* fail *if* $\mathsf{Authorized}(\mathsf{vk}_{\mathsf{SA}}, \mathsf{sid}, \mathsf{vk}_{\mathsf{sid}}, \mathsf{id}_\ell) = \mathsf{fail}$ *or if* A *has queried* $\mathsf{Submit}(1^n, \mathsf{sid}, \cdot, m_j, cred_{\mathsf{id}_\ell})$ *for either* $\ell, j \in \{0, 1\}$
- *Let* $\mathsf{Sub}_\ell = \mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{vk}_{\mathsf{sid}}, m_\ell, cred_{\mathsf{id}_{\ell\oplus b}})$ *for both* $\ell = 0$ *and* $\ell = 1$ *and finally output* $A(1^n, (\mathsf{Sub}_0, \mathsf{Sub}_1), z')$

1) Justification for the definition: We want to allow the adversary to participate in multiple surveys, with multiple honest submitters, and see the submissions of as many honest submitters as it wishes. In particular, we can consider a definition in which the following changes are made. The adversary is now permitted to register an unlimited number of honest users by interacting with Reg^U under the condition that no two have the same id. The adversary, for an arbitrary k surveys and $\ell \geq 2$ honest submitters in each survey, is allowed to output two $k \times \ell$ matrices of ids ID^0,ID^1, a vector of k sids, $\vec{\mathsf{sid}}$, a vector of k the surveys' verification-keys $\vec{\mathsf{vk}}_{\mathsf{sid}}$ and two $k \times \ell$ matrices of messages M^0, M^1. We require the submissions be legitimate, thus we require that all of the ids in a given row of ID^c, $c \in \{0, 1\}$ be distinct. Similarly, all the sids in $\vec{\mathsf{sid}}$ must be distinct, corresponding to distinct surveys (if one wanted to have more ids in a given survey, the adversary could simply increase the size of ℓ). Finally, all of the surveys need to be "well formed" for every $i \in [k], j \in [\ell]$, both $\mathsf{ID}^0_{i,j}$ and $\mathsf{ID}^1_{i,j}$ are authorized to participate in survey sid_i. Finally, of course, we do not permit the attacker to query the Submit oracle for submissions by $\mathsf{id}_{i,j}$ in survey sid_i). We call this notion *multi-survey unlinkability*. We note that this is a form of concurrent security as it guarantees unlinkability no matter how the adversary generates multiple surveys and schedules the registration of individuals. In the full version of this paper, we formally define this notion and show that it is implied by Definition 3.

2) Security: Let us now turn to define security. The following definition stipulates that only authorized users may complete surveys, and only one of their submissions is counted. We require that this holds even if the user attacker may register multiple identities, and see submissions of the attacker's choice for any other user of its choice and in any survey (this one, or any other survey).

To begin, we formalize what it means to give the attacker access to submission oracles for users of its choice by defining the stateful oracle $\mathsf{Submit}'(1^n, \mathsf{sid}, pk_{\mathsf{sid}}, m, \mathsf{id}, \mathsf{pk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}})$ that operates as follows: if the oracle has not previously been queried on the identity id, let $(out_{\mathsf{RA}}, cred_{id}) \leftarrow (\mathsf{Reg}^{\mathsf{RA}}(\mathsf{sk}_{\mathsf{RA}}, 1^n, \mathsf{pk}_{\mathsf{RA}}, \mathsf{id}), \mathsf{Reg}^U(1^n, \mathsf{pk}_{\mathsf{RA}}, \mathsf{id}))$; next output $\mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, m, cred_{\mathsf{id}})$. If the oracle has previously been queried on the identity id, recover the previously computed credential $cred_{\mathsf{id}}$, and directly output $\mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, m, cred_{\mathsf{id}})$.

Definition 4: An ad-hoc survey scheme Γ is *secure against malicious users* if for every non-uniform PPT A, every polynomial $p(\cdot)$, there exists a negligible function $\mu(\cdot)$, such that the following experiment outputs success with probability at most $\mu(n)$ for every $n \in \mathbb{N}$,

MU(A, n)

- $(\mathsf{vk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}}) \leftarrow \mathsf{GenRA}(1^n)$
- *For* $i = 1$ *to* $p(n)$, *let* $(\mathsf{vk}^i_{\mathsf{SA}}, \mathsf{sk}^i_{\mathsf{SA}}) \leftarrow \mathsf{GenSA}(1^n)$
- Throughout the rest of the experiment, A has access to oracle $\mathsf{GenSurvey}(1^n, \cdot, \cdot, \mathsf{sk}^i_{\mathsf{SA}})$ for any i, and $\mathsf{Submit}'(1^n, \cdot, \cdot, \cdot, \mathsf{vk}_{\mathsf{RA}}, \mathsf{sk}_{\mathsf{RA}})$.
- Let adversary $A(1^n)$ concurrently interact with $\mathsf{Reg}^{\mathsf{RA}}(\mathsf{sk}_{\mathsf{RA}}, 1^n)$ with adaptively chosen, but unique, user identities $id \in \{0, 1\}^n$ of its choice. (That is, A can only use each user identity id in a single interaction with $\mathsf{Reg}^{\mathsf{RA}}$.) Let L' denote the list of user identities selected by A, and let z denote the output of A after this interaction finishes.

– $z', L, \mathsf{sid}, i \leftarrow A(1^n, z)$
– $\mathsf{vk}_{\mathsf{sid}} \leftarrow \mathsf{GenSurvey}(1^n, \mathsf{sid}, L, \mathsf{sk}^i_{\mathsf{SA}})$
– $S \leftarrow A(1^n, z', \mathsf{vk}_{\mathsf{sid}})$
– *Output* success *if*
- $|S| > |L \cap L'|$ *and*
- $\mathsf{Check}(pk_{\mathsf{SA}}, pk_{\mathsf{RA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, \mathsf{Sub})$ *accepts for all* $\mathsf{Sub} \in S$,
- *For* $(\mathsf{tok}, m, \mathsf{tokauth})$, $(\mathsf{tok}', m', \mathsf{tokauth}') \in S^2$, $\mathsf{tok} \neq \mathsf{tok}'$.
- *For all* $(\mathsf{tok}, m, \mathsf{tokauth}) \in S$, $(\mathsf{tok}, m, \mathsf{tokauth}')$ *was never received as an output from* A*'s* Submit' *oracle (where* $\mathsf{tokauth}'$ *is an arbitrary string) when queried with* sid *as second input.*

These four conditions roughly correspond to the determining whether A produced more submissions than allowed, all submissions are valid, all submissions have different token-numbers, all token-numbers are new, and no submissions have been modified.

Remark: Note that in the above definition, A is allowed to talk to $p(n)$ different SAs. It is without loss of generality to assume that A talks to just a single SA (that is $p(n) = 1$). This follows from standard technique, and is omitted for space reasons.

III. An Ad-hoc Survey Scheme Based on General Assumptions

We assume the reader is familiar with *signature schemes* secure against adaptive-chosen message attacks [26], non-interactive *commitment schemes* [27], and *pseudorandom functions* [28]; see [29].

A. Concurrent simulation extractable NIZK

We introduce the new notion of *tag-based concurrent simulation-extractable (cSE) NIZK* (a non-interactive version of tag-based concurrent simulation-extractable zero-knowledge from [14], [15]); this notion is closely related to the notion of *non-malleability in the explicit witnesses sense* of [30] (which in turn relies on the notion of *simulation-soundness* of [13]), and *universally composable UC NIZK* of [9]. The former is (a-priori) weaker than ours in that it only requires extraction from a single protocol (this notion is referred to as "many-one" simulation-extractability in [14], [15]) whereas the latter is stronger in that it requires extractability to be done "on-line". Relying on this new intermediate notion allows us to strike the right balance between security and efficient implementability: in particular, we will present simple and extremely efficient concurrent simulation-extractable NIZKs in the Random Oracle model, whereas UC NIZK incurs more overhead.

We first start by defining concurrent simulation extractability. Imagine an attacker A playing man-in-the-middle between a legitimate prover *on the left interaction*, and a legitimate verifier *on the right*. More specifically, in the left interaction, attacker A can request proofs of any true statement x of its choice using any tag tag of its choice. In the right interaction, A outputs a list of tags, statements and proofs $(\vec{\mathsf{tag}}, \vec{x}, \vec{\pi})$ as well as a string aux. For every such A, we require the existence of a simulator-extractor SE that must reconstruct the view of A and additionally produce witnesses for all accepting statement-proofs $(x, \pi) \in (\vec{x}, \vec{\pi})$ on the right that use a new tag.

Below, we assume a proof system (D, P, V) where D generates a CRS, P is a prover, and V is a verifier. The function W provides witnesses to theorem statements. Let $real(1^n, A, W, z)$ denote the output of the following experiment: Run $\rho \leftarrow D(1^n)$; next give $A(1^n, z, \rho)$ oracle access to $P'(\cdot, \cdot, \cdot)$ where $P'(\mathsf{tag}, x, \rho)$ runs $P(\mathsf{tag}, x, W(x, \mathsf{view}'), \rho)$ where view' is A's view up until this query; finally, outputs the view of A (that is a sequence of tag-statement-proof tuples $(\mathsf{tag}, \vec{x}, \pi)$ and the CRS ρ. Given a view $view$ of A, we interpret the output of A as a sequence of tag-statement-proof tuples $(\mathsf{tag}', x', \pi')$ and some additional output aux. Define the predicate $\mathsf{fail}(view, \vec{w}) = 1$ if and only if A, when given the view $view$, containing the CRS ρ, outputs a proof π of some statement x with respect to tag tag such that a) none of the proofs received by A in the view $view$ use tag tag, b) V accepts the proof π—that is, $V(\mathsf{tag}, x, \rho, \pi) = 1$, and c) $\vec{w}$ does not contain a witness for x.

Definition 5 (Concurrent Simulation-Extractability): Let (D, P, V) be a non-interactive proof system for the language L. We say that (D, P, V) is *concurrently simulation extractable (cSE)* if for every PPT A, there exists an expected PPT simulator-extractor SE such that the following two conditions hold:

(Simulatability) For every witness function $W(\cdot, \mathsf{view}')$ such that $W(x, \mathsf{view}') \in R_L(x)$ for all $x \in L$ and all view', the following ensembles are computationally indistinguishable

- $\{real(1^n, A, S, W, z)\}_{n \in \mathbb{N}, z \in \{0,1\}^*}$
- $\{(view, \vec{w}) \leftarrow SE(1^n, z) : view\}_{n, z \in \{0,1\}^*}$

(Extractability) There exists a negligible function μ such that for every $n \in \mathbb{N}$,

$$\Pr[(view, \vec{w}) \leftarrow SE(1^n, z) : \mathsf{fail}(view, \vec{w}) = 1] \leq \mu(n)$$

Additionally, we say that (D, P, V) is *black-box concurrently simulation extractable* if there exists an expected PPT oracle-machine $\tilde{SE}$ such that for every A, the above two conditions hold with respect to $SE(1^n, z) = \tilde{SE}^{A(1^n, z, \cdot)}(1^n)$.

1) Simulation extractability in the ROM: The definition of tag-based non-interactive arguments in the ROM is identical to Definition 5 except that there

is no need for procedure D; instead of sampling $\rho \leftarrow D(1^n)$, we sample ρ as random function from $\{0,1\}^{\mathrm{poly}(n)} \rightarrow \{0,1\}^{\mathrm{poly}(n)}$ for some appropriate polynomial; furthermore, instead of providing ρ as input to P and V, both algorithms have oracle access to ρ. We make the analogous change in the definition of the real experiment in the definition of simulation extractability (now additionally, A gets oracle access to ρ instead of getting it as input). Also, the view of A contains all of the answers to oracle calls to ρ made by A, and thus the simulator-extractor SE needs to reconstruct those (as opposed to reconstructing the whole of ρ).

Simulation-extractability from HVZK: In this section we show how to transform any 3-round special-sound special Honest-verifier Zero-knowledge (HVZK) [31] proof/argument into a black-box concurrently simulation-extractable NIZK in the ROM.

Definition 6 ([31]): A proof system (P, V) is a *3 round special-sound Honest-verifier Zero-knowledge (HVZK) proof/argument for binary relation* R if:

1) (three-move form) Let α be the common input to P and V, and β such that $(\alpha, \beta) \in R$ is private input to P. (P, V) has the following three-move form:
   a) P sends a message a to V.
   b) V sends a random t-bit challenge string c.
   c) P replies z, and V accepts if $\phi(\alpha, a, c, z) = 1$ for some polynomial-time predicate ϕ.
2) (special soundness) Given accepting transcripts (a, c, z) and (a, c', z') for the instance α such that $c \neq c'$, there exists a polynomial-time algorithm computing β such that $(\alpha, \beta) \in R$.
3) (special honest-verifier ZK) There exists a polynomial time simulator Sim, which on input α and a random challenge c, outputs an accepting conversation of the form (a, c, z), which is identically distributed to transcript generated by $P(\alpha, \beta)$ and $V(\alpha)$ for any $(\alpha, \beta) \in R$.

Given a protocol Π that is a 3-round special-sound Honest-verifier Zero-knowledge (HVZK) proof/argument for binary relation R, we obtain the tag-based NIZK $\tilde{\Pi}$ for R by applying the Fiat-Shamir heuristic to Π and additionally requiring that the tag tag is hashed—that is, the second-message challenge c is generated by applying the random oracle ρ to the first message a and the tag tag (i.e., $b = \rho(a, \mathsf{tag})$).

Theorem 1: Let Π be a special-sound special HVZK argument for L, where the first message a of Π has $\omega(\log n)$ min-entropy [8] and the second message b is of length $\omega(\log n)$. Then $\tilde{\Pi}$ is a tag-based black-box concurrently simulation-extractable argument.

Proof: (Omitted for space.) ■

[8]Every special-sound special HVZK argument for a hard-on-the-average language must have this property: if not, with polynomial probability two honest executions would have the same first message, but different second messages and thus a witness can be extracted out.

B. The construction

Assuming familiarity with basic cryptographic primitives discussed in the previous section, the construction is easy to understand at a high-level. We assume that all users have unique string identifiers, e.g. email addresses, to identify them in the protocol. The RA and SA are each have keys for digital signatures which are considered the output of GenRA and GenSA respectively.

A user's secret credential is a random string s. The user generates the credential by generating a commitment to the random string s and proving the correctness of the commitment to the RA in zero-knowledge. During registration, the RA verifies that this proof corresponds to the correct identity of the user and then authorizes the credential by signing the commitment and the user id with its signing key. The security of the signature scheme ensures that only the RA can authorize legitimate credentials. The signature on the commitment can be interpreted as a very weak form of blind-signature in which the RA does not learn anything about the values being signed, but can verify that the user has knowledge of the underlying secret.

To generate a survey, an SA generates a unique sid and individually signs the pair (id, sid) for each user id that can participate in the survey. We call the list of all generated signatures L. Only those IDs which are signed by the SA will be able to complete the given survey, and the unforgeability properties of the signature scheme ensure that only the SA can modify the list. Beneficially, SAs do not need to interact with any other players to create the surveys.

To submit a response to a survey, the user sends her submission m to the SA along with an NIZK proof that the she has a valid credential (i.e., she has a commitment signed by the RA, and she knows the values committed to), and that her credential corresponds to an identity on the survey list L (i.e., the ID in her signed commitment is also in signed in L). Only legitimate users can submit in the scheme, and clearly the submission is anonymous to the SA who learns nothing except that the submitter is on its approved list. However, we need to achieve two other properties: i) tie the submission m to the proof and the survey id, and ii) ensure that the submitter does not submit multiple responses. The first property prevents submissions from replay and is achieved because the NIZK we use is tag-based; we can use the sid concatenated to the message m as the tag for the proof. To prevent multiple submissions, we include a unique token number created by evaluating a pseudo-random function on the sid using the user's

credential secret s as the seed, and we augment the NIZK proof to show that the unique token number is computed correctly and corresponds to the user's credential. Thus, every valid submission by a given user will have the same random token number associated with it. This token does not reveal any information about the submitter, but it allows the SA to detect multiple submissions from the same anonymous user and either discard all of them, or accept only the latest submission.

1) Primitives used: Let,

- $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme.
- $\{f_s\}_{s\in\{0,1\}^*}$ be a family of PRFs.
- Com be a commitment scheme.
- Let L_1 be the NP language defined as follows: $(1^n, c) \in L_1$ iff there exists strings $r \in \{0,1\}^*, s \in \{0,1\}^n$ such that $c = \mathsf{Com}(s; r)$.
- Let L_2 be the NP language defined as follows: $(\mathsf{tok}, \mathsf{sid}, \mathsf{pk}_{\mathsf{RA}}, \mathsf{pk}_{\mathsf{SA}}) \in L_2$ iff there exist strings $s, id, c, r, \sigma_s, \sigma_{\mathsf{sid}_{id}}$ such that $c = \mathsf{Com}(s; r)$ and $\mathsf{Ver}_{\mathsf{pk}_{\mathsf{RA}}}(c||id, \sigma_s) = 1$ and $\mathsf{Ver}_{\mathsf{pk}_{\mathsf{SA}}}(\mathsf{sid}||id, \sigma_{\mathsf{sid}_{id}}) = 1$ and $\mathsf{tok} = f_s(\mathsf{sid})$
- Let (D_1, P_1, V_1) and (D_2, P_2, V_2) be black-box cSE NIZK protocols for L_1 and L_2 respectively.

2) Abstract ad-hoc survey scheme Γ:

- $\mathsf{GenRA}(1^n) = \mathsf{Gen}(1^n)$, $\mathsf{GenSA}(1^n) = \mathsf{Gen}(1^n)$.
- $(\mathsf{Reg}^{\mathsf{RA}}(\mathsf{sk}_{\mathsf{RA}}), \mathsf{Reg}^{U})(1^n, id_i)$ proceeds as follows:
  - A user with identity id uniformly generates and stores $s \leftarrow \{0,1\}^n, r \leftarrow \{0,1\}^{\mathrm{poly}(n)}$ and computes $c = \mathsf{Com}(s; r)$.
  - i computes a cSE-NIZK π using P_1 (given the CRS or RO) that $(1^n, c) \in L_1$, using the tag 0^n and using (s, r) as witness.
  - The user sends c, id, π to the RA.
  - The RA checks that π with tag 0^n is an accepting proof that $(1^n, c) \in L_1$ (using V_2) and if so returns $\sigma_s = \mathsf{Sign}_{\mathsf{sk}_{\mathsf{RA}}}(c||\mathsf{id})$; otherwise it simply returns fail.
  - The user outputs $cred = (c, s, r, \sigma_s)$.
- $\mathsf{GenSurvey}(1^n, \mathsf{sid}, L, \mathsf{sk}_{\mathsf{SA}})$ proceeds as follows. For each $id \in L$, compute $\sigma_{\mathsf{sid}_{id}} = \mathsf{Sign}_{\mathsf{SA}_{\mathsf{sk}}}(\mathsf{sid}||id)$ and output the list of tuples $(id, \sigma_{\mathsf{sid}_{id}})$.
- $\mathsf{Authorized}(\mathsf{pk}_{\mathsf{SA}}, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, id)$ outputs YES if $\mathsf{pk}_{\mathsf{sid}}$ contains a record of the form $(id, \sigma_{\mathsf{sid}_{id}})$ such that $\mathsf{Ver}_{\mathsf{SA}_{\mathsf{pk}}}(\mathsf{sid}||id, \sigma_{\mathsf{sid}_{id}}) = 1$
- $\mathsf{Submit}(1^n, \mathsf{sid}, \mathsf{pk}_{\mathsf{sid}}, m, id, cred)$ proceeds as:
  - Parse $cred = (c, s, r, \sigma_s)$.
  - Compute token-number $\mathsf{tok} = f_s(\mathsf{sid})$.
  - Recover a tuple of the form $(id, \sigma_{\mathsf{sid}_{id}})$ from $\mathsf{pk}_{\mathsf{sid}}$. If $\mathsf{Ver}_{\mathsf{SA}_{\mathsf{pk}}}(\mathsf{sid}||id, \sigma_{\mathsf{sid}_{id}}) \neq 1$, abort.
  - Compute a cSE NIZK π using P_2 (and the CRS or RO) that $(\mathsf{tok}, \mathsf{sid}, \mathsf{pk}_{\mathsf{RA}}, \mathsf{pk}_{\mathsf{SA}}) \in L_2$ with tag $1||\mathsf{sid}||m$ using $c, s, r, \sigma_s, \sigma_{\mathsf{sid}_{id}}$ as witness.
  - Send the tuple $\mathsf{Sub} = (\mathsf{tok}, m, \pi)$ to the SA.
- $\mathsf{Check}(\mathsf{pk}_{\mathsf{RA}}, \mathsf{pk}_{\mathsf{SA}}, \mathsf{sid}, \mathsf{tok}, m, \pi)$ outputs accept if V_2 (given the CRS or RO) accepts π as a proof of the statement $(\mathsf{tok}, \mathsf{sid}, \mathsf{pk}_{\mathsf{RA}}, \mathsf{pk}_{\mathsf{SA}}) \in L_2$ with tag $\mathsf{sid}||m$.

Theorem 2: If $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is a secure signature scheme, and $\{f_s\}_{s\in\{0,1\}^*}$ is a family of secure PRFs, and Com is a perfectly hiding, computationally binding commitment scheme, and (D_1, P_1, V_1) and (D_2, P_2, V_2) are cSE NIZKs for the languages L_1, L_2 respectively, then scheme Γ is a multi-survey unlinkable (Def. 3), ad-hoc survey scheme that is secure (against malicious users) (Def. 4).

Proof: (Omitted for space.) ■

C. Alternative implementations

Our proof for theorem 2 applies to more general implementations of the Reg protocol as well. In particular, the proof allows for the user's output from the Reg^{U} protocol to be a signature on $s||id$ instead of $\mathsf{Com}(s)||id$ as long as the signature scheme remains unforgeable even when the user is given access to a $\mathsf{Reg}^{\mathsf{RA}}(\mathsf{sk}_{\mathsf{RA}})(\cdot)$ oracle. Recall that the standard security property of a signature scheme allows the adversary oracle access to the $\mathsf{Sign}(\cdot)$ function; providing oracle access to $\mathsf{Reg}^{\mathsf{RA}}$ instead is a natural generalization. This generalization allows for more efficient concrete implementations. In particular, it allows one to exploit natural connections between specifically engineered commitment schemes and signature schemes.

IV. Concrete Instantiation

A. Bilinear Groups and Assumptions

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p. A *symmetric bilinear map* is an efficient mapping $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ which is both: (*bilinear*) for all $g \in \mathbb{G}$ and $a, b \leftarrow \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; and (*non-degenerate*) if g generates $\mathbb{G}$, then $e(g, g) \neq 1$. A more general version of the bilinear map is the *asymmetric bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct groups. We present our constructions in symmetric groups (which are easier for readers to parse), but then conduct our implementation in asymmetric groups (which are more efficient).

Assumption 1 (Decisional Bilinear Diffie-Hellman): Let g generate a group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$ with an efficient bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. For all non-uniform PPT adversaries A, the following probability is negligible in λ:

$$\left| 1/2 - \Pr\left[\begin{array}{ll} a, b, c \leftarrow \mathbb{Z}_p; x \leftarrow \{0,1\}; & \\ T_0 \leftarrow e(g,g)^{abc}; & \\ T_1 \leftarrow \mathbb{G}_T; & : x = x' \\ x' \leftarrow \mathrm{A}(g, g^a, g^b, g^c, T_x) & \end{array} \right] \right| .$$

Roughly this assumption states that an adversary cannot distinguish between $e(g,g)^{abc}$ and a random group element when given (g, g^a, g^b, g^c).

Assumption 2 (n-Decisional Diffie-Hellman Inversion): Let h generate a group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. For all non-uniform PPT adversaries A, the following probability is negligible in λ:

$$\left| 1/2 - \Pr \left[\begin{array}{l} b \leftarrow \mathbb{Z}_p^*; x \leftarrow \{0,1\}; \\ T_0 \leftarrow h^{1/b}; T_1 \leftarrow \mathbb{G}; \\ x' \leftarrow \mathrm{A}(h, h^b, h^{b^2}, \ldots, h^{b^n}, T_x) : \\ \quad x = x' \end{array} \right] \right|.$$

B. Scheme

The common input for all protocols is a description of the bilinear mapping, together with generators g, h of $\mathbb{G}$, and a description of a CRHF H that maps $\{0,1\}^* \to \mathbb{Z}_q$. The values g, h can be chosen randomly by the RA. The elliptic curve library that we use implements the hash operation H differently depending on the curve implementation.

Our scheme makes use of the Pedersen commitment scheme [27], the Dodis-Yampolskiy pseudo-random function [32], and a simplified signature scheme derived from the Boneh-Boyen IBE [33]; all three are summarized in Fig. 1.

Theorem 3 ([27]): The Pedersen commitment scheme is a perfectly-hiding and computationally-binding commitment scheme assuming the hardness of the discrete logarithm problem in $\mathbb{G}$.

Theorem 4 ([32], [12]): (informal) The Dodis-Yampolskiy PRF is a secure pseudo-random function for input space $\mathbb{Z}_q^*$ in the generic group model.[9]

Theorem 5 ([33]): (informal) Under the Decisional Bilinear Diffie-Hellman assumption in $\mathbb{G}$, the Boneh-Boyen scheme is adaptively secure for n bit messages for "large groups" which can withstand a factor of $1/2^n$ reduction in security. [10]

C. Sigma protocols for languages L_1 and L_2

In order to instantiate the remaining protocols, we must specify the languages L_1 and L_2 defined in the abstract section and provide cSE NIZKs for those languages. Recall that languages L_1 and L_2 implicitly depend on the specification of a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, a commitment scheme Com, and a pseudo-random function family $\{f_s\}$. For the rest of these sections, assume that these three dependencies are instantiated with the BB-signature scheme, the Pedersen commitment, and the Dodis-Yampolskiy function as described above in Fig. 1. We now provide Σ-protocols for L_1, L_2 and then apply the Fiat-Shamir heuristic in the random oracle model as prescribed in Thm.1 to produce the required cse-NIZK.

The language L_1 corresponds to a standard Schnorr-like [34] proof for knowledge of a representation of a discrete logarithm. In the Camenish-Stadler notation [35], such a protocol is specified as follows:

$$L_1 = PoK\left\{(s_{id}, d) : \alpha = v^{s_{id}} g^d\right\}$$

This denotes a "zero-knowledge proof of knowledge of integers s_{id}, d such that $\alpha = v^{s_{id}} g^d$ holds" where α, v, g are elements of some group G. Values not in the parentheses are considered to be public.

(1) COMMITMENT SCHEME

Common Input: An algebraic group $\mathbb{G}$ of prime order q with generators g, h.

Commit(m) $S \longrightarrow R$ Sender chooses random $s \in Z_q$, and sends $\alpha = g^m h^s$ to R.

Open(α)**:** $S \longrightarrow R$ Sender sends (m, s) to R. Receiver checks $\alpha \stackrel{?}{=} g^m h^s$.

(2) DODIS-YAMPOLSKIY PRF F_y THAT MAPS $Z_q^* \to \mathbb{G}_t$ WHERE $y \in Z_q$.

Setup: An algebraic group $\mathbb{G}$ of prime order q with generator g and PRF seed $y \in Z_q$.

$F_y(m)$**:** The user computes $F_y(m) = g^{1/(y+m)}$ for any $m \neq 0 \mod q$.

(3) BB SIGNATURE SCHEME

$\mathsf{Gen}(1^n)$: Sample the secret key $\mathsf{sk} \leftarrow \alpha \in \mathbb{Z}_q$. Sample random group generators u, v, g, h and compute $U = e(g,g)^\alpha$. The verification key is $\mathsf{vk} \leftarrow (u, v, g, h, U)$.

$\mathsf{Sign}(\mathsf{sk}, m_0, m_1)$**:** Choose $r \in \mathbb{Z}_q$ randomly and compute

$$\sigma_1 \leftarrow g^\alpha (u^{m_0} v^{m_1} h)^r, \quad \sigma_2 \leftarrow g^r$$

and output (σ_1, σ_2) as the signature.

$\mathsf{Ver}(\mathsf{vk}, m_0, m_1, \sigma_1, \sigma_2)$**:** Accept if

$$e(\sigma_1, g) \stackrel{?}{=} U \cdot e(u^{m_0} v^{m_1} h, \sigma_2)$$

Figure 1. A commitment, and PRF family, and signature scheme

[9] Dodis and Yampolskiy [32] showed that their PRF was secure under the (parameterized) Decisional-Diffie-Hellman Inversion Assumption assumption in $\mathbb{G}$ for "small inputs"; they discuss [32, Section 4.3] how the input to their PRF can be 160 bits for proper choice of parameters and that arbitrarily-long strings can first be hashed down to 160 bits using any collision-resistant hash function.

[10] Suppose that messages of n bits are signed using the scheme, and recall that one can always sign messages of arbitrary length by first applying a collision-resistant hash function to map them to n bits. Then the adversary's advantage in forging a message increases by a factor of 2^n. Thus, to be provably-safe in using this scheme in applications that demand full security, one has to choose their parameters carefully. Boneh and Boyen suggest, as one example, that when $n = 160$ and the bilinear group is set so that no polynomial-time adversary can break DBDH with advantage 2^{-240}, then the resulting signature scheme offers roughly 80-bit security.

Theorem 6 ([34]): There exists a 3-round honest-verifier special-sound zero-knowledge protocol for L_1.

In contrast, the language L_2 is more complicated to specify and requires a non-trivial protocol. A statement in this language consists of the tuple $(\mathsf{sid}, C, \mathsf{pk}_{\mathsf{RA}}, \mathsf{pk}_{\mathsf{SA}})$ where $\mathsf{pk}_{\mathsf{RA}} = (u, v, h, e(g,g)^x)$ and $\mathsf{pk}_{\mathsf{SA}} = (u_v, v_v, h_v, e(g,g)^y)$. The witness for an instance is the tuple $(s_{\mathsf{id}}, \mathsf{id}, c, r, \sigma, \sigma_{\mathsf{sid}_{id}})$ such that $\sigma = (\sigma_1, \sigma_2)$ forms a Boneh-Boyen signature on the values $(\mathsf{id}, s_{\mathsf{id}})$, $\sigma_{\mathsf{sid}_{id}} = (\sigma_{\mathsf{sid}_{id},1}, \sigma_{\mathsf{sid}_{id},2})$ forms a Boneh-Boyen signature on $(\mathsf{sid}, \mathsf{id})$, and $C = F_{s_{\mathsf{id}}}(\mathsf{sid})$ where F is the Dodis-Yampolskiy PRF, for the signature schemes above.

In the first step of the proof for L_2, the prover re-randomizes $(\sigma, \sigma_{\mathsf{sid}_{id}})$ by choosing random $d_1, d_2 \in \mathbb{Z}_q$ and computes

$$\begin{aligned}(s_1 = \sigma_1 \cdot (u^{\mathsf{id}} v^{s_{\mathsf{id}}} h)^{d_1} \quad &, \quad s_2 = \sigma_2 \cdot g^{d_1}) \\ (s_3 = \sigma_{\mathsf{sid}_{\mathsf{id}},1} \cdot (u_v^{\mathsf{sid}} v_v^{\mathsf{id}} h)^{d_2} \quad &, \quad s_4 = \sigma_{\mathsf{sid}_{\mathsf{id}},2} \cdot g^{d_2}).\end{aligned}$$

The values s_2, s_4 are sent to the Verifier, and the problem reduces to proving a simpler statement: (a) (s_1, s_2) form a Boneh-Boyen signature on the values $(\mathsf{id}, s_{\mathsf{id}})$, (b) (s_3, s_4) form a Boneh-Boyen signature on $(\mathsf{sid}, \mathsf{id})$, and (c) $C = F_{s_{\mathsf{id}}}(\mathsf{sid})$ as follows:

$$PoK \left\{ \begin{array}{l} (\mathsf{id}, s_{\mathsf{id}}, s_1, s_3) : \\ \quad E^x e(h, s_2) = e(s_1, g) e(u^{\mathsf{id}} v^{s_{\mathsf{id}}}, s_2)^{-1} \wedge \\ \quad E^y e(u_v^{\mathsf{sid}} h_v, s_4) = e(s_3, g) e(v_v^{id}, s_4)^{-1} \wedge \\ \quad E \cdot C^{-\mathsf{sid}} = C^{s_{\mathsf{id}}} \end{array} \right\}$$

where $E = e(g,g)$.

The Σ-protocol (P_2', V_2') for this simpler language proceeds as follows:

1) $P_2' \rightarrow V_2'$ Prover picks random $b_1, b_2 \in \mathbb{Z}_q$ and $J_1, J_2 \in G$ and computes

$$\begin{aligned} E_1 &\leftarrow e(J_1, g) \cdot e(u^{b_1} v^{b_2}, s_2)^{-1} \\ E_2 &\leftarrow e(J_2, g) \cdot e(v_v^{b_2}, s_4)^{-1} \\ E_3 &\leftarrow C^{b_2} \end{aligned}$$

2) $P_2' \leftarrow V_2'$ Verifier picks a random $c \in \mathbb{Z}_q$.

3) $P_2' \rightarrow V_2'$ Prover computes a response

$$\begin{aligned} z_1 &\leftarrow b_1 + c \cdot id & z_2 &\leftarrow b_2 + c \cdot s_{id} \\ z_3 &\leftarrow s_1^c \cdot J_1 & z_4 &\leftarrow s_3^c \cdot J_2 \end{aligned}$$

4) Verifier checks the following:

$$\begin{aligned} E_1 \cdot e(g,g)^{xc} \cdot e(h, s_2)^c &= e(z_3, g) \cdot e(u^{z_1} v^{z_2}, s_2)^{-1} \\ E_2 \cdot e(g,g)^{yc} \cdot e(v_v^{\mathsf{sid}} h_v, s_4)^c &= e(z_4, g) \cdot e(v_v^{z_1}, s_4)^{-1} \\ E_3 \cdot e(g,g)^c \cdot C^{-c(\mathsf{sid})} &= C^{z_2} \end{aligned}$$

Theorem 7: The above (P_2', V_2') is an honest-verifier special-sound zero-knowledge protocol for L_2.

Proof: (sketch) The completeness of the protocol is standard. First we show honest-verifier zero-knowledge. On input an instance and a random challenge c, the simulator first chooses a random $z_1, z_2 \in \mathbb{Z}_q$ and random $z_3, z_4 \in G$ and computes

$$\begin{aligned} E_1 &= \frac{e(z_3, g) \cdot e(u^{z_1} v^{z_2}, s_2)^{-1}}{e(g,g)^{cx} \cdot e(h, s_2)^c} \\ E_2 &= \frac{e(z_4, g) \cdot e(v_v^{z_1}, s_4)^{-1}}{e(g,g)^{cy} \cdot e(v_v^{\mathsf{sid}} h_v, s_4)^c} \\ E_3 &= \frac{C^{z_2}}{e(g,g)^c \cdot C^{-c(\mathsf{sid})}} \end{aligned}$$

and outputs $(E_1, E_2, E_3), c, (z_1, z_2, z_3, z_4)$ as the transcript. By inspection, it follows that the distribution of transcripts is perfectly identical to a transcript from a successful protocol execution.

We now show that the protocol is special-sound. Consider two transcripts $(E_1, E_2, E_3), c, (z_1, z_2, z_3, z_4)$ and $(E_1, E_2, E_3), c', (z_1', z_2', z_3', z_4')$ where $c \neq c'$ that both pass the verification test. It follows that

$$\begin{aligned} id &= \frac{z_1 - z_1'}{c - c'} \\ s_{id} &= \frac{z_2 - z_2'}{c - c'} \\ s_1 &= \left(\frac{z_3}{z_3'}\right)^{c - c'} \\ s_3 &= \left(\frac{z_4}{z_4'}\right)^{c - c'} \end{aligned}$$

since both transcript tuples satisfy the three equations in Step 4) of the Σ-protocol. ■

Corollary 8: In the random oracle model, there exists a BB cse NIZK for languages L_1 and L_2 for the signature schemes $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, commitment scheme Com and PRF $\{F_s\}$ describe above.

Proof: Follows from Thm. 1. ■

D. GenRA *and* GenSA *protocols*

The GenRA and GenSA methods are the key generation methods for the BB signature scheme. More specifically, the RA picks random group elements $u, v, h \in G$ and a secret element $x \in \mathbb{Z}_q$. The RA's public key $\mathsf{RA}_{vk} = (u, v, h, e(g,g)^x)$ and $\mathsf{RA}_{\mathsf{sk}} = x$. (RA will be signing m_1 as the id with u and m_2 as the user's secret seed with v.)

The SA picks random group elements $u_v, v_v, h_v \in G$ and a secret element $y \in \mathbb{Z}_q$. The SA's public key $\mathsf{SA}_{vk} = (u_v, v_v, h_v, e(g,g)^y)$ and $\mathsf{SA}_{\mathsf{sk}} = y$. ($m_1$ will be the sid and m_2 will be the user id of a participant authorized to submit in the survey.)

E. The Reg protocol

Common: Group (G, e, g), $\mathsf{RA}_{vk} = (u, v, h, e(g,g)^x)$

RA Secret Key: x

User identity: id

User and RA establish a mutually authenticated secure communication channel.

$U \to \mathsf{RA}$ The user chooses a random PRF seed $s_{id} \in \mathbb{Z}_q$ and a random $d \in \mathbb{Z}_q$, computes $\alpha = v^{s_{id}} g^d$, and sends (id, α) to RA.

The user also gives a (NI)zero-knowledge proof of knowledge for $(s_{id}, d) \in L_1$ using the Σ-protocol for L_1 described above:

$$PoK\left\{(s_{id}, d) : \alpha = v^{s_{id}} g^d\right\}$$

$U \to \mathsf{RA}$ User picks a random b_1, b_2 and sends $\gamma = v^{b_1} g^{b_2}$ to RA.

$U \to \mathsf{RA}$ User generates a random challenge $c \in \mathbb{Z}_q$ by using the random-oracle H and the tag 0^n as $c = H(g, \mathsf{RA}_{vk}, id, \alpha, \gamma, 0^n)$

$U \to \mathsf{RA}$ User computes $z_1 = b_1 + c s_{id}$, $z_2 = b_2 + cd$ and sends (z_1, z_2) to RA.

RA verifies $v^{z_1} g^{z_2} \stackrel{?}{=} \alpha^c \gamma$.

$U \longleftarrow \mathsf{RA}$ RA checks that the identity id has not been registered before. RA chooses $r \in \mathbb{Z}_q$ randomly, computes the signature tuple $\sigma_1 \leftarrow g^x (u^{id} \alpha h)^r, \sigma_2 \leftarrow g^r$ and sends R the signature $\sigma_{id} = (\sigma_1, \sigma_2)$.

U User verifies the signature by checking that

$$e(\sigma_1, g) = e(g,g)^x \cdot e(u^{id} v^{s_{id}} g^d h, \sigma_2).$$

If this verifies, the user removes the commitment randomness by computing $\sigma_1' = \sigma_1 / \sigma_2^d$ and stores the secret credential $(id, s_{id}, \sigma_{id} = (\sigma_1, \sigma_2))$.

Note that at the end of this protocol, the user stores a signature on $s||id$ under the verification key $\mathsf{RA_{vk}}$ (instead of a signature on $c||id$ where c is a commitment as per the abstract protocol). As mentioned earlier in section III-C, this choice of implementation preserves security as long as the signature scheme remains unforgeable even if the adversary has oracle access to the $\mathsf{Reg}^{\mathsf{RA}}$ function. In this case, it is easy to show that unforgeability holds by showing how to simulate the $\mathsf{Reg}^{\mathsf{RA}}$ function given access to a signing oracle. At a high level, the simulation works by (a) using the simulator-extractor to extract $(s_{\mathsf{id}}, \mathsf{id})$ from the NIZK proof that the user provides, and then submitting a query for a signature (σ_1, σ_2) on $s||id$ to the signature oracle, and finally, multiplying an extra g^{dr} term into the σ_1 term to produce messages for the adversary that are indistinguishable from ones received in the given interaction. The fact that the user must provide a cse-NIZK with a tag 0^n that is different from all other tags used in all other protocol instances allows simulation-extraction to function and thus enables a proper simulation.

F. Survey Registration

SA Input: $\mathsf{SA}_{vk} = (u_v, v_v, h_v, e(g,g)^y)$, $\mathsf{SA_{sk}} = y$, $\mathsf{sid} \in \mathbb{Z}_q$

List of identies: L

SA For each $id \in L$, the SA computes the following: Pick a random $r \in \mathbb{Z}_q$ and compute

$$\sigma^{\mathsf{sid}_{id}} = (g^y (u_v^{\mathsf{sid}} v_v^{id} h)^r, g^r)$$

Publish the list $L_{\mathsf{sid}} = (\mathsf{sid}, \{id_i, \sigma^{\mathsf{sid}_{id}}\}_{i \in L})$

Authorized**:** Anyone can verify that a user with identity id is authorized to submit in survey sid by finding the corresponding signature $\sigma^{\mathsf{sid}_{id}} = (\sigma_1, \sigma_2)$ in L_{sid} and then checking that

$$e(\sigma_1, g) \stackrel{?}{=} e(g,g)^y \cdot e(u_v^{\mathsf{sid}} v_v^{id_i} h, \sigma_2).$$

G. Submission

The Submit protocol is instantiated using the Σ-protocol implementation for the L_2 language.

Common Input: (G, e, g), the list L_{sid}, the public keys $\mathsf{SA}_{vk} = (u_v, v_v, h_v, e(g,g)^y)$, and $\mathsf{RA}_{vk} = (u, v, h, e(g,g)^x)$

User Secrets: id, submission m, credential (σ_{id}, s_{id})

The user aborts if the user has already participated in an survey with sid or $\mathsf{sid} = s_{id}$. The user and SA establish a secure connection in which SA is authenticated, but the user is anonymous.

U The user identifies the tuple $(\mathsf{sid}, id_i, \sigma^{(i)})$ in L_{sid} in which $id_i = id$. The user computes $F_{s_{id}}(\mathsf{sid}) = C \leftarrow e(g,g)^{1/(s_{id} + \mathsf{sid})}$.

$U \to \mathsf{SA}$ User sends $(\mathsf{sid}, C, m, s_2, s_4)$ and an NIZKPOK of the statement (id, s_{id}, s_1, s_3) in L_2 with tag $1||\mathsf{sid}||m$ to the SA:

SA : If the proof verifies, record the submission (C, m) replacing any prior occurrence of $(C, \cdot)$.

Theorem 9 (Security of the Survey System): Assuming the security of the Pedersen commitment scheme, the Dodis-Yampolskiy PRF for input space $\mathbb{Z}_q^*$, the adaptive security of the Boneh-Boyen signatures, and a collision-resistant hash function, the above concrete instantiation is a correct (Def. 2) ad-hoc survey scheme that is (multi-survey) unlinkable (Def. 3) and secure against malicious users (Def. 4) in the random oracle model.

Proof: Security of the each of the primitives follows from Thm. 5, Thm. 4, Thm. 3, Thm. 6, Thm 7, and Cor 8. The rest then follows from Thm. 2. ■

V. Implementation of Concrete Scheme

Because practicality and efficiency were major goals, the concrete instantiation of the system was implemented in C++11 using the MIRACL big number library [36], which provides support for pairing based cryptography and is free for educational purposes. We implemented with curves that MIRACL equates to 128 AES security, using a Barreto-Naehrig pairing friendly curve, with embedding degree k=12, and the Ate pairing. The implementation uses an asymmetric pairing, which follows immediately from our protocol.[11]

We analyzed performance for both types of curves because large groups are necessary for the survey system to be provably-secure based on "standard" bilinear assumptions (e.g. DBDH). In particular, for the reduction to standard assumptions to work, the Dodis-Yampolskiy PRF needs a group much larger than its input size and the Boneh-Boyen signature reduction uses a complexity leveraging argument that also requires a large group size. However, one can also analyze these building blocks, and thus the larger system, in the generic group model [37], which provides evidence that the scheme is secure against generic attacks and is then traditionally implemented with smaller groups.

Our implementations, which are not particularly optimized, show efficiency that is more than sufficient for all practical surveys. In particular, our implementation utilizes only 1 core of the CPU; it is straightforward to parallelize user registration, and survey verification over multiple cores and machines by simply having all cores run the same processes and balancing the load (i.e., the number of registrations or surveys to verify) given to any particular core. Similarly, when generating new surveys, we can split the participant list among a number of different cores at the SA, and each would sign the names of the individuals on its portion of the list.

Our results show that one or two modern workstations or server systems are sufficient to manage surveys into the millions using the more efficient smaller sized groups, and a small number of high-performance machines (on the order of 5 to 10) would easily handle surveys of larger sizes or similar sizes using the larger group size. User side computation is reasonably negligible. Submitting a survey, or verifying a submitted survey, the most expensive operations a user might want to do, took in the strongest security setting at most 2.5 seconds.

1) System: All tests were done on a 3.06 GHZ Intel Core 2 Duo, Late 2009 iMac with 12GB 1067 MHZ DDR3 RAM with a 5400RPM SATA HD. This machine is several years old and much slower than a modern server.

A. Experiments

When considering survey life cycles, there are three actions that are potentially computationally intensive: i) mass registration of users, ii) generation of large surveys by the SA, and iii) verifying ballots for large surveys, as many ballots may need to be verified in real time over a short time period. Generating RA keys and SA keys is computationally efficient on our system, and moreover they are done once per entity and unlikely to be generated in large quantity. Similarly, so long as submitting and user registration are not so slow as to cause consternation, their performance is relatively unimportant, as each user performs their own computation in a decentralized manner. In contrast, large surveys may be generated and run in a centralized location, so it is important that the generation of the survey list be scalable, and a reasonable system be able to validate a large number of incoming submitters.

We performed the following experiments i) RA Key Generation, ii) SA Key Generation, iii) User Registration, iv) survey Generation by the SA, and v) Submission. RA and SA generation is simple, and we simply ran the protocols. For user generation we constructed a large set of unique user names, and registered each user sequentially. We report on the time taken per registration. We recorded the computation time for the user and the RA separately. For survey registration we took the user-names generated previously, and constructed survey lists out of them. Since this is one large computation, we report on the aggregate time for a small survey of 300 submitters, and the average time per submission. We have verified that this time scales linearly with the number of submitters as one would expect.[12] Finally, we consider actual submission, and measure the time for the submitter to submit their response, and the time necessary for the SA to verify the submission. These measurements are done per submitter.

Each of the experiments below was performed 100 times, with mean and standard deviation of times reported in milliseconds in Table I. The measured times correspond only to the time necessary to compute the appropriate cryptography and store the result to disk. There is no network measurements involved. We discuss this in the next subsection. The most expensive operation is the mass verification of surveys that should be done by a survey authority when surveys are submitted,

[11] During the user registration protocol, the verifier sends back another element $\sigma_3 = g^r$, so that the user can compute $\sigma_1' = \sigma_1/\sigma_3^r$. This is the only extra information needed to accommodate the asymmetry, and clearly does not affect security.

[12] We have created surveys of 1 million users on our machine, but due to time constraints only computed this once with the smaller group: it took approximately 42 minutes, inline with the expected linear extrapolation.

Table I
TIMING RESULTS FROM THE IMPLEMENTATION OF OUR CONCRETE SYSTEM.

Operation	BN Curve		BLS Curve	
	Mean (ms)	StdDev (ms)	Mean (ms)	StdDev (ms)
RA Key Gen	55.30	2.69	882.94	147.41
SA Key Gen	14.54	1.93	224.21	60.23
User Side User Registration	3.35	0.71	6.11	18.67
RA Side User Registration	7.91	2.36	13.65	30.09
User Verification User Registration	58.25	25.62	69.69	103.49
SA survey Generation (300 submitters)	706.85	16.47	8,116.11	911.62
SA survey Generation (per Questnr.)	2.36		27.05	
User Submission	88.20	4.97	1,482.98	144.02
SA Verify Submission	121.52	7.03	2,247.29	251.28

to ensure their legitimacy. In the more practical smaller group sized implementation (i.e., the less stringent security assumptions) we can verify 1 million submissions in about 33 hours per core on our system. Assuming a reasonable 4 cores per system gives us a little over 8 hours for 1 system. Or 3 systems could process in about 2 hours. Even in the most stringent security case, assuming we had to verify the submissions of 1 million people, we could use about 20 machines with 4 cores each, and compute the results in under 8 hours. If there is no need to keep the survey results private, this computing power can be rented from the cloud (e.g., AWS), making the costs low. Verification does not need private information, so there is less risk in renting resources. Survey generation and the RA's side of user registration are other places where computing costs are centralized with an authority. Both are at least an order of magnitude less time intensive than survey verification, and can be distributed over similar resources efficiently.

Storage and Bandwidth Requirements: Storage and bandwidth requirements are both very reasonable for such schemes. Each element in the survey list output during the Survey Registration is less thank 1KB, as are the users' secret tokens. The most expensive NIZK used in the submission of the survey is smaller than 8KB. The above excludes the length of the IDs, which are system dependent, but are reasonably on the order of a few hundred bytes at most.

B. Anonymous Communication & Participant Lists

In practice, the user needs to anonymously submit a single message to the SA during survey submission. In moderate-security settings, proxy services can be used to transmit the data, and in high-security settings, onion-routing such as TOR [38] may be used. Another issue that arises is the distribution of the survey participant list for a survey. For small surveys, this is inconsequential, but when participant rolls get into the millions, the file of eligible submitters with corresponding information can become large. Deploying this to each user is definitely feasible (a typical OS patch push hits millions of people), but there are easy alternatives that can slacken the requirement. E.g., wild-cards can be used to ease enrollment, or separate participant lists can be constructed of smaller size: Anonymity is slightly weakened, but we are not aware of any surveys where participants scale to a million submitters with full anonymity, and thus some weakening may be acceptable.

ACKNOWLEDGMENTS

All opinions expressed and implied in this work are solely those of the authors and do not represent or reflect the views of their respective universities.

REFERENCES

[1] S. Staff, "Security breach leaves 45,000 at risk of identity theft," 2009. [Online]. Available: http://thetruthwillrise.wordpress.com/2009/06/25/security-breach-leaves-45000-at-risk-of-identity-theft/

[2] P. C. A. for Submission., "Observations on the Course Review System at the University of Virginia," 2013.

[3] M. Riley, "U.s. agencies said to swap data with thousands of firms," 2013. [Online]. Available: http://www.bloomberg.com/news/2013-06-14/u-s-agencies-said-to-swap-data-with-thousands-of-firms.html

[4] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *CRYPTO*, vol. 740, 1992, pp. 89–105.

[5] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth," in *Eurocrypt 1995*, 1995.

[6] A. Neff, "A verifiable secret shuffle and its application to e-voting," in *CCS 2001*, 2001.

[7] J. Benaloh, "Simple verifiable elections," in *EVT 2006*, 2006.

[8] B. Adida, "Helios: Web-based open-audit voting," in *USENIX 2008*, 2008.

[9] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *FOCS '01*, 2000, see updated version at Cryptology ePrint Archive: Report 2000/067.

[10] A. Juels, M. Luby, and R. Ostrovsky, "Security of blind digital signatures (extended abstract)," in *CRYPTO '97*, 1997, pp. 150–164.

[11] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *CRYPTO*, 2004, pp. 56–72.

[12] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich, "How to win the clonewars: Efficient periodic n-times anonymous authentication," in *ACM CCS '06*, 2006, pp. 201–210.

[13] A. Sahai, "Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security," in *FOCS'99*, 1999, pp. 543–553.

[14] R. Pass and A. Rosen, "Concurrent non-malleable commitments," *SIAM Journal of Computing*, 2008.

[15] ——, "New and improved constructions of non-malleable cryptographic protocols," *SIAM Journal of Computing*, 2008.

[16] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO '86*, 1986, pp. 186–194.

[17] D. Chaum and E. van Heyst, "Group signatures," in *EUROCRYPT '91*, 1991, pp. 257–265.

[18] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *EUROCRYPT*, 2003, pp. 614–629.

[19] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO '04*, 2004, pp. 45–55.

[20] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *ASIACRYPT '01*, 2001, pp. 552–565.

[21] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *CT-RSA*, 2005, pp. 136–153.

[22] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete." *Communications of the ACM*, vol. 28(10), pp. 1030–1044, October 1985.

[23] J. Camenisch and A. Lysyanskaya, "Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation," in *EUROCRYPT '01*, vol. 2045, 2001, pp. 93–118.

[24] ——, "A signature scheme with efficient protocols," in *SCN*, 2002, pp. 268–289.

[25] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in *EUROCRYPT '05*, 2005, pp. 302–321.

[26] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Computing*, vol. 17(2), pp. 281–308, 1988.

[27] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO*, 1991, pp. 129–140.

[28] O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[29] O. Goldreich, *The Foundations of Cryptography*. Cambridge University Press, 2001.

[30] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Robust non-interactive zero knowledge," *SIAM Journal on Computing*, vol. 20, pp. 1084–1118, 2001.

[31] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *CRYPTO*, 1994, pp. 174–187.

[32] Y. Dodis and A. Yampolskiy, "A Verifiable Random Function with Short Proofs and Keys," in *PKC '05*, vol. 3386 of LNCS, 2005, pp. 416–431.

[33] D. Boneh and X. Boyen, "Efficient selective-ID secure Identity-Based Encryption without random oracles." in *EUROCRYPT '04*, 2004, pp. 223–238.

[34] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptography*, vol. 4, pp. 161–174, 1991.

[35] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *CRYPTO '97*, vol. 1296 of LNCS, 1997, pp. 410–424.

[36] M. Scott, "Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL)," published by Shamus Software Ltd., http://www.shamus.ie/.

[37] V. Shoup, "Lower bounds of discrete logarithms and related problems," in *Proceedings of Eurocrypt '97*, 1997, pp. 256–266.

[38] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX 2004*, 2004.