An Overview of 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

November 26, 2014

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. This technology has numerous applications including university course evaluations, online product reviews, whistleblowing and more.

We overview our recent progress on ad-hoc surveys, including a discussion of the security and privacy properties required by such a system and results from a prototype implementation called ANONIZE. Our performance analysis shows that ANONIZE enables securely implementing million-person anonymous surveys using a handful of modern workstations. 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, and is practical enough for use today.

1 Introduction

Companies, universities, health providers and government agencies often attempt to collect data from targeted groups of users by running surveys. Such surveys aim to satisfy two basic, but conflicting properties: survey results need to be *authentic* (i.e., only a specific set of users should be allowed to submit in the data collections and each user should only be allowed to submit once), yet surveys must also be *anonymous* (i.e., there should not be a link between the legitimate user and his/her submitted survey data so that users feel safer about submitting honest feedback). The most straightforward way to implement authenticity is for the survey implementor to request usernames during submission, but this obviously breaks user anonymity. The most straightforward way to implement anonymity is to avoid collecting usernames during submission, but this may allow attacks in which a malicious user submits hundreds of responses to skew the result.

One way to overcome this conflict is to employ a trusted third party to collect usernames during submission (to guarantee authenticity), but to then delete usernames when providing the survey results to the survey initiator. Placing such trust in a survey collector, however, may be too 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. 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 which users have 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.

Furthermore, in many situations, jurisdictional boundaries or legal requirements make it unfeasible to rely on trusted third parties to maintain anonymity: 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. Finally, if a trusted third party removes all identifying information with a submission in order to provide anonymity or accepts submissions from anonymized networks, then the trusted party loses the ability to verify if a participant submits multiple responses.

In light of these deficiencies, we seek cryptographic solutions to the problem of anonymous surveys that provide security guarantees in which anonymity and authenticity hold *without needing to trust a third party*.

Cryptographic voting techniques offer a partial solution to this problem (starting with [1]). 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, one day, the ease-of-use of the survey is significantly hampered and becomes 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.

1.1 Our innovation: electronic ad-hoc surveys

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; thus, it is expected to increase user participation and make the feedback submitted more valuable.

We describe an ad-hoc survey scheme, ANONIZE, and report on an implementation of the protocol that admits practical and efficient solutions for very large surveys. A proof of security for the cryptographic protocols in ANONIZE can be found in [2]; this proof holds even if the attacker(s) participate in an arbitrary number of concurrent surveys. 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.

[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.

2 Ad-Hoc Surveys

The Parties and Steps In more details, there are three parties in an ad-hoc survey system: a *single* registration authority (RA) that issues master user tokens, one or more survey authorities (SA) that can create surveys, and multiple 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 generating a unique 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 entirely on its own without interaction with either the RA or the users in the system. Finally, a user who is on the list of valid identities for a survey can submit a response to the survey by simply routing one message to the SA (through an anonymous network like Tor, or anonymous proxy relay).

Once all the submissions have been collected, the SA may (depending on external privacy requirements) publish a list of all survey responses. If the survey responses are made public, then they can be audited. A survey responder can check that her submission was "counted" by simply inspecting the submission output. Moreover, there is a procedure for anyone to check that each submission was from a unique authorized user (e.g., user can check that there is no "ballot/survey-stuffing") and the list of authorized users.

The Applications We anticipate that this technology could apply to many online survey scenarios and possibly enable new ones. To illustrate how these surveys might be used, we provide three detailed examples.

2.1 A University Course Evaluation Example

It is standard practice for most universities to allow the students to evaluate each course that they complete. These surveys typically include write-in sections where open-ended answers are encouraged. In the past, many universities conducted these surveys on papers handed out and then collected during one of the final classes. Increasingly, many universities are moving to online surveys to increase participation and ease data collection. Currently, a link to an online course evaluation survey is typically emailed to each student, who then must trust the website collecting their response to keep them anonymous. As we discussed in the introduction, this is a dangerous assumption, even if the website makes a good faith effort to do so. To increase student confidence in the system, we now describe how an ad-hoc survey would function in this setting.

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 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*.

Course Survey Setup Whenever an administrator wishes to set-up a course survey, she generates a *survey key* based on a unique survey ID she chooses, such as "Survey for CS350 for Spring 2014 by Professor Brown at Cornell University", and a list of the email addresses of the course participants (e.g., sarah@cornell.edu, mark@cornell.edu, ratna@cornell.edu, etc.).

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 unlinkable *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. If they are announced, the student can verify that her response was included.

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).

2.2 An Online Product Review Example

Many online retailers today display a set of customer reviews next to each product. These reviews are often influential to prospective customers. To avoid returns and customer dissatisfaction, these retailers have a vested interest in the credibility of these reviews. To bolster this credibility, many retailers are indicating which reviewers it can verify purchased this product on their site (e.g., see Amazon Verified Purchase). This is currently implemented in a non-anonymous fashion, meaning the retailer knows exactly which customer posted which review. We conjecture that a significant fraction of the customers would be more likely to post a review if their review could be anonymous. We now explore how ad-hoc surveys can give users this anonymity, while still allowing the retailer to verify their purchase (and letting them post at most one review per purchase).

Customer Account Creation When a user creates her online account with a retailer (providing an email address and credit card to confirm her identity), the user can be given the option to enroll in anonymous reviewing. If she chooses to do so, she can then interact with the retailer to generate an unlinkable master user token that is tied to her online account identifier (e.g., username or email address). This step can also be done at a later stage if the user desires or loses her credential, but it only needs to be done once.

Product Purchase Transaction Whenever a transaction is made involving a user enrolled in anonymous reviewing, the retailer can add this user's online account identifier to an (internal) certified list of purchasers of a given product. This list together with a product identifier form the survey ID. (This list is actually a collection of signatures. The retailer can give the signature certifying that user's purchase to the user *and not the entire purchase list* if the retailer wishes to keep the other customers' purchases private.)

Review Execution If the user wishes to post a review for a product she purchased, the user's client (either computer or smart phone) combines the survey ID (or just the portion of it containing her purchase) and her master user token to generate an unlinkable one-time token that she can use to complete the review. This token carries no link to the user's identity, but can only be used once. Once the retailer receives the review with the token (which could be routed anonymously), the retailer can verify it and post it as a "Verified Purchase Review".

Again, once Step 1 has been done (the user enrolls in anonymous reviewing and obtains his master use token), an unlimited number of purchases and reviews can be performed.

In the above, we have the retailer helping to create the master user tokens. Alternatively, users could obtain their master user tokens by interacting with their bank or credit card company. Retailers, restaurants, service providers, etc. could then generate lists of authorized reviewers based on the bank account number

[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.

or credit card number used for the purchase. The review execution would then remain the same. This model could work well for websites (e.g., Yelp) that review other parties services, because they could guarantee that they were only posting reviews of actual customers, while the customers themselves could remain anonymous.

2.3 Whistleblowing

Frequently a whistleblower will wish to provide information to the ombudsman of an organization about alleged misconduct that the whistleblower believes has occurred. Further, due to fears of reprisal, many whistleblowers prefer to remain anonymous. However upon receiving a complaint for investigation, an ombudsman generally needs to first ascertain that the source of the complaint is legitimate, say, from a verified employee of the organization and not just sent by a random discontent. In many cases a whistleblower may be able to prove that they are in the organization by providing information that only an employee would know, but doing so often decreases or removes their anonymity. We now show how an ad-hoc survey can give whistleblowing employees anonymity while still allowing an ombudsman to verify that a complaint comes from within the organization.

Employee Account Creation When an employee first joins an organization, they are registered and issued a master user token tied to their system account (e.g., username or email address). Concurrently, the ombudsman adds them to a "whistleblowing" survey which consists of all employees in the organization, and provides the employee –in conjunction with the master user token– a signature showing that their employee ID is certified as a participant for the "whistleblowing" survey. The ombudsman also publishes a signed list of all participants (i.e., all employees) to show that all employees can participate.

Whistleblowing and Verification Should an employee uncover some illegal or unethical activities they can write a memo to the ombudsman via the "whistleblowing" survey. They certify the memo with their master user token and signed employee ID on the"whistleblowing" participant list. Upon receipt (via anonymous channel), the ombudsman can verify that the submission comes from a valid survey participant, and thus a legitimate employee.

2.4 Features and Security Requirements

In a nutshell, there are two crucial aspects of an ad-hoc survey. The first 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. This privacy property primarily benefits users, although the surveyor may benefit from increased participation and reduced motivation to bias a response. The *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*. The security property primarily benefits surveyors, although the user is also assured that his response will not be lost in a deluge of unauthorized responses (e.g., only customers who actually purchased the product can post a review of it.)

Part of our contribution in [2] is to precisely define security properties of ad-hoc surveys. 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 [3]. 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.

In a game-based definition, there are two parties: a challenger (who represents all honest parties) and an adversary (who represents all corrupted parties). The challenger and adversary interact with each other according to the rules of the game, e.g., the adversary may be able to ask to register corrupted users and to see the survey outputs generated by honest users of his choice. At some point in the game, the challenger gives the adversary a challenge, e.g., an honestly generated survey response. At the end of the game, the adversary provides a response to this challenge, e.g., the adversary might guess which honest user generated the challenge. If the adversary's response to the challenge is correct, he is said to have won the game. The definition of security states that for any realistic time-bounded adversary's probability of winning the game is very close to the probability of winning based on a random guess. Thus, a proof under this definition rules out *all* realistic attackers, provided that the game accurately captures all the actions that the adversary can make in the real world. The formalization of both the security definitions and the corresponding proofs require real care, especially for a system as complex as ad-hoc surveys.

This is analogous with other cryptographic game-based definitions, e.g., blind signatures [4]; we emphasize that although related notions of anonymity and authenticity have been defined in the literature for other applications, such as group signatures, ring signatures and anonymous credentials, our setting is considerably more complex and thus the actual definitions are different.

Privacy/Unlinkability. The first important property is *survey unlinkability*. Informally, the organizer of the survey (SA) and the RA should not be able to link users to their survey responses by analyzing the message traffic of the protocol (separate measures are taken to ensure network-layer unlinkability). We require that this holds even if the 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).

We introduce an adaptive notion of unlinkability in which the survey responses remain unlinkable even if the adversaries (the SA and RA) force other users to submit submissions in a certain way at various points during the security experiment. Roughly speaking, the game starts by the adversary establishing the parameters for one RA. Two distinct honest users id_0 and id_1 "register" with the adversarial RA. The adversary then outputs the public information for a "challenge survey" and he receives back the survey submissions for *both* id_0 and id_1 (in a random order). The adversary is said to win, if he can correctly guess which submission was formed by which user. The definition states that no realistic time-bounded adversary can win with probability much better than $1/2$.

Security/Authenticity. The second important property is *authenticity*, namely, malicious users should not be able to submit responses to a survey unless they are authorized by the SA to participate in the survey. This property should also hold when the user arbitrarily creates fake identities in the system, fake surveys, and new surveys that are "related" to the survey that the user is attempting to attack. Moreover, this property ensures that a (potentially) malicious user can only complete such surveys once (or more precisely, if they successfully submit multiple times, their submissions all use the same token-number and can be easily identified and joined, or discarded depending on the survey policy).

Roughly speaking, the security game starts by the challenger establishing the parameters for one RA and many SAs. The attacker is then allowed to (1) generate new survey IDs for any SA, (2) ask for any survey submissions output by any honest user for surveys of his choice, and (3) register corrupted users. The challenge involves a new survey ID honestly generated, but where the attacker was allowed to choose both the list L of users and the SA. The attacker makes a response of a set of survey submission and these submissions are checked against five conditions where breaking any one results in the attacker winning. These five conditions roughly correspond to determining whether the attacker produced more submissions than allowed, all submission are valid, all submissions have different token-numbers, all token-numbers are new, and no submissions have been modified. The definitions states that no realistic time-bounded adversary

can win with probability much better than 0.

3 Building ANONIZE

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]. A commitment scheme allows the Sender to commit to a message without revealing that message to the Receiver. A signature scheme allows for public authentication of a message. A pseudorandom function is a deterministic function that maps any input to a "random-looking" output. Finally, a NIZK provides a proof of an assertion (e.g., I know a signature by the RA on message m) without revealing *anything* beyond the truth of this statement (e.g., does not reveal any bits of the signature). We prove the security of the abstract scheme based on the assumption that all generic primitives employed are secure. 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 a *specific* commitment scheme, signatures scheme, PRF and NIZKs to obtain our efficient secure ad-hoc survey scheme ANONIZE. This system is now based on specific computational assumptions related to the security of the underlying primitives. As in many other efficient cryptographic systems, our analysis treats the hash function (used as part of the NIZK) as if it were an idealized "random oracle" [5]. 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, e.g., [6], although our constructions differ in a few ways. As far as we know, our scheme is also one of the first implementations of a complex cryptographic scheme that is concurrently-secure.

Step One: An Abstract Construction. 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; see [7] for more).

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".

Survey To create a survey, an SA generates a new signing key and publishes a list of signed user identities along with the survey id/verification key, v_{id}.

Response To complete a survey for survey id v_{id}, a user id generates a *single-use token* $F_{s_{\mathsf{id}}}(v_{\mathsf{id}})$ (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 $F_{s_{\mathsf{id}}}(v_{\mathsf{id}})$. 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

[4] As we show in [2], we actually need a new variant of standard NIZKs.

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 [8] and *universally composable UC NIZK* of [3]). 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, e.g., [9]) 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.

Step Two: A Concrete Construction. 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 [10].[6] 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. This integration step is non-trivial as we have to look closely at the underlying algebraic structure of each building block to find primitives that can leverage a common algebraic structure to result in efficient NIZKs and thereby produce an overall efficient system.

While we provide *one* method of implementing the abstract system, there are endless alternatives. We chose this concrete implementation based on its efficiency, simplicity, and our confidence in the security of the underlying building blocks.

Our ANONIZE construction is placed in a "bilinear map" algebraic setting, which is commonly believed to provide high security levels for a relatively smaller group size (compared to, say, RSA), which can translate into low-bandwidth and computationally-efficient implementations [11]. These bilinear maps are typically implemented via an underlying elliptic curve. The common input for all protocols is a description of the bilinear mapping, together with generators for the algebraic groups involved and a description of a collision-resistant hash function H. The bilinear map we chose is described further in Section 4 and will typically be chosen as one of a handful of options from a given library. The elliptic curve library that we use implements the hash function H differently depending on the curve implementation. The generators can be chosen randomly by the RA.

With these common settings established, our system then makes use of the following building blocks: the Pedersen commitment scheme [12], the Dodis-Yampolskiy pseudo-random function [13], and a simplified signature scheme derived from the Boneh-Boyen identity-based crypto system [14]. The final non-trivial step was to devise the efficient NIZKs for the statements we need to prove concerning these building blocks. We defer this deeper technical discussion to [2].

4 An Implementation and Experiments

ANONIZE can easily handle large numbers of users with moderate resources for the registration and survey authorities. 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 well at costs that are easily affordable.

[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 $F_{s_{\mathsf{id}}}(v_{\mathsf{id}})$ value. The old survey with this value can be deleted.

[6] The Random Oracle Model is a very practical one, used in most deployed cryptographic protocols.

Table 1: 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 users)	706.85	16.47	8,116.11	911.62
SA survey Generation (per user)	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

We tested our system by implementing it in C++11 using the MIRACL big number library [15], which provides support for pairing (i.e., bilinear map) based cryptography and is free for educational purposes. We implemented with two curves using the Ate pairing: a Barreto-Naehrig (BN) pairing friendly curve that MIRACL equates to 128 bit security and a Barreto-Lynn-Scott (BLS) pairing friendly curve that MIRACL equates to 256 bit security. Depending on how aggressive of a security assumption one is willing to make, we get either close to 128 and 256 bit security from these implementations, or we get 128-bit security from the 256 bit implementation, and the 128-bit implementation would not be secure–due to possible lose of security in the security proof. 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.

Our implementations, which are not particularly optimized, show efficiency that is more than sufficient for nearly all practical surveys. In particular, our implementation utilizes only one 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 workstations or servers are sufficient to manage surveys into the millions using the more efficient BN curves, 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 BLS curves. 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 at worst 2.5 seconds.

4.1 Experiments

Table 1 shows the results of our experiments. Each of the experiments below was performed 100 times, with mean and standard deviation of times reported in milliseconds. 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. The most expensive operation is the mass verification of surveys that should be done by a survey authority. In the BN implementation, 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. In the BLS setting, 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 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 than 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.

References

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

[2] S. Hohenberger, S. Myers, R. Pass, and abhi shelat, "ANONIZE: A Large-Scale Anonymous Survey System," in *IEEE Security and Privacy*, 2014.

[3] 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.

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

[5] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS '03*, 1993, pp. 62–73.

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

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

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

[9] R. Pass and A. Rosen, "New and improved constructions of non-malleable cryptographic protocols," *SIAM Journal of Computing*, 2008.

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

[11] D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil Pairing," in *CRYPTO '01*, vol. 2139 of LNCS, 2001, pp. 213–229.

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

[13] 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.

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

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