The Phantom Tollbooth: Privacy-Preserving Electronic Toll Collection in the Presence of Driver Collusion

Sarah Meiklejohn* *UC San Diego*
Keaton Mowery† *UC San Diego*
Stephen Checkoway‡ *UC San Diego*
Hovav Shacham§ *UC San Diego*

Abstract

In recent years, privacy-preserving toll collection has been proposed as a way to resolve the tension between the desire for sophisticated road pricing schemes and drivers' interest in maintaining the privacy of their driving patterns. Two recent systems in particular, VPriv (USENIX Security 2009) and PrETP (USENIX Security 2010), use modern cryptographic primitives to solve this problem. In order to keep drivers honest in paying for their usage of the roads, both systems rely on unpredictable spot checks (e.g., by hidden roadside cameras or roaming police vehicles) to catch potentially cheating drivers.

In this paper we identify large-scale driver collusion as a threat to the necessary unpredictability of these spot checks. Most directly, the VPriv and PrETP audit protocols both reveal to drivers the locations of spot-check cameras — information that colluding drivers can then use to avoid paying road fees. We describe Milo, a new privacy-preserving toll collection system based on PrETP, whose audit protocol does not have this information leak, even when drivers misbehave and collude. We then evaluate the additional cost of Milo and find that, when compared to naïve methods to protect against cheating drivers, Milo offers a significantly more cost-effective approach.

1 Introduction

Assessing taxes to drivers in proportion to their use of the public roads is a simple matter of fairness, as road maintenance costs money that drivers should expect to pay some part of. Gasoline taxes, currently a proxy for road use, are ineffective for implementing congestion pricing for city-center or rush-hour traffic. At the same time, the detailed driving records that would allow for such congestion pricing also reveal private information about drivers' lives, information that drivers do seem to have interest in keeping private. (In the U.S., for example, some courts have recognized drivers' privacy interests by forbidding the police from using a GPS device to record a driver's movements without a search warrant [1].)

The VPriv [38] and PrETP [4] systems for private tolling, proposed at USENIX Security 2009 and 2010 respectively, attempt to use modern cryptographic protocols to resolve the tension between sophisticated road pricing and driver privacy. At the core of both these systems is a monthly payment and audit protocol. In her payment, each driver commits to the road segments she traversed over the month and the cost associated with each segment, and reveals the total amount she owes. The properties of the cryptography used guarantee that the total is correct assuming the segments driven and their costs were honestly reported, but that the specific segments driven are still kept private.

To ensure honest reporting, the systems use an auditing protocol: throughout the month, roadside cameras occasionally record drivers' locations; at month's end, the drivers are challenged to show that their committed road segments include the segments in which they were observed, and that the corresponding prices are correct. So long as such spot checks occur unpredictably, drivers who attempt to cheat will be caught with high probability given even a small number of auditing cameras. In the audit protocols for both VPriv and PrETP, however, *the authority reveals to each driver the locations at which she was observed.* (The driver uses this information to open the appropriate cryptographic commitments.) If the cameras aren't mobile, or are mobile but can be placed in only a small set of suitable locations (e.g., overpasses or exit signs along a fairly isolated highway), then the drivers will easily learn where the cameras are (and, perhaps more importantly, where they aren't). Furthermore, if drivers collude and share the locations at which they were challenged, then a few audit periods will suffice for colluding drivers to learn and map the cameras' locations.

*smeiklej@cs.ucsd.edu
†kmowery@cs.ucsd.edu
‡s@cs.ucsd.edu
§hovav@cs.ucsd.edu

We believe the model of large-scale driver collusion is a realistic one. For example, drivers already collaborate to share the locations of speed cameras [36] and red-light cameras [37]; if we extend this behavior to consider maps of audit cameras, then we see that the unpredictable spot checks required in the analysis of VPriv and PrETP are difficult to achieve in the real world when drivers may collude on a large scale. When drivers know where cameras are (and where they aren't), they will not pay for segments that are camera-free, and may even change driving patterns to avoid the cameras. By collaborating, drivers can discover and share camera locations at acceptable cost; in fact, if the cameras are revealed to them directly in the course of the audit protocol then they can do so without incurring a single fine.

Finally, one might argue that an appropriate placement of audit cameras at chokepoints will make them impossible to avoid, even if their location is known; the price charged for traversing such a chokepoint could then be made sufficiently high that it subsidizes the cost of maintaining other, unaudited road segments. This alternative arrangement may seem superficially appealing, but it is ultimately incompatible with driver privacy. If drivers cannot avoid a chokepoint they cannot but be observed by authorities when they cross it; in other words, this approach would be feasibly enforceable only when most drivers are regularly observed at the chokepoints. In fact, what we have described is precisely the situation today in many cities, where tolls are collected on bridges and other unavoidable chokepoints.

Our contribution We show, in Section 4, how to modify the PrETP system to obtain our own system, Milo, in which the authority can perform an agreed-upon number of spot checks of a driver's road-segment commitments without revealing the locations being checked. To achieve this, we adapt a recent oblivious transfer protocol due to Green and Hohenberger [28] that is based on blind identity-based encryption. We have implemented and benchmarked our modifications to the audit protocol, showing (in Section 5) that they require a small amount of additional work for each driver and a larger but still manageable amount of work for the auditing authority.

Basic fairness demands that drivers whom the authority accuses of cheating be presented with the evidence against them: a photo of their car at a time and location for which they did not pay. This means that drivers who intentionally incur fines will inevitably learn some camera locations; in some cases, a large coalition of drivers may therefore profitably engage in such misbehavior. Here the information about camera locations is leaked not by the audit protocol but by the legal proceedings that follow it.

Finally, if the cameras are themselves visible then drivers will discover and share their locations, regardless of the cryptographic guarantees of the audit protocol.[1] All that is necessary is for one driver to spot the camera at any point during the month; the colluding drivers can then ensure that their commitments take this camera into account. We discuss this further in Section 6.

In summary, our paper makes three concrete contributions:

- we identify large-scale driver collusion as a realistic threat to privacy-preserving tolling systems;
- we modify the PrETP system to avoid leaking camera locations to drivers during challenges; and
- we identify and evaluate other ways to protect against driver collusion and compare their costs to that of Milo.

2 System Outline

In this section we present an overview of the Milo system. We discuss both the organizational structure of the system, as well as the security goals it is able to achieve. As our system is built directly on top of PrETP we have approximately maintained its structure, with the important differences highlighted below.

2.1 Organization

Milo consists of three main players: the driver, represented by an On-Board Unit (OBU); the company operating the OBU (abbreviated TSP, for Toll Service Provider); and finally the local government (or TC, for Toll Charger) responsible for setting the road prices and collecting the final tolls from the TSP, as well as for ensuring fairness on the part of the driver. The interactions between these parties can be seen in Figure 1.

In some respects, the organization of Milo is similar to that of current toll collection systems. The driver will keep a certain amount of money in an account with the TSP; at the end of every month the driver will then pay some price appropriate for how much she drove and the amount of money remaining in the account will need to be replenished. The major difference, of course, is that the payments of the driver do not reveal any information about their actual locations while driving.[2] In addition, we will require that the TC perform occasional spot checks to guarantee that drivers are behaving honestly.

The OBU is a box installed in the car of the driver, which is responsible for collecting location information, computing the prices associated with the roads, and forming the final payment information that is sent to the TSP

[1] De Jonge and Jacobs [19] appear to have been the first to note that unobservable cameras are crucial for random spot checks.

[2] As also noted by Balasch et al. [4], the pricing structure itself may of course reveal driver locations — e.g., if segment i costs 2^i (see Section 4), then all drivers' paths are revealed by cost alone. This will likely not be a problem in practice.

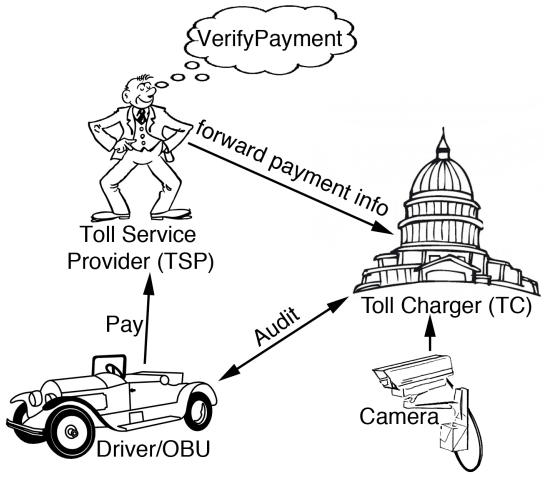

Figure 1: An overview of how the Milo system works. As we can see, the OBU deals with the TSP for payment purposes (using the Pay protocol), but for spot checks it interacts with the TC (using the Audit protocol). The TC conducts these audits using both the information recorded by the cameras it operates along the roads and the OBU's payment information, which is forwarded on from the TSP after it has been checked to be correct (using the VerifyPayment protocol).

at the end of each month. Its work in this stage is described formally in our Pay algorithm, which we present in Section 4.

The TSP is responsible for the collection of tolls from the driver. At the end of each month, the TSP will receive a payment message from the OBU as specified above. It is then the job of the TSP to verify that this payment information is correct, using the VerifyPayment algorithm outlined in Section 4. If the payment information is found to be correctly formed then the TSP can debit the appropriate payment from the user's account; otherwise, they can proceed in a legal manner that is similar to the way in which traffic violations are handled now.

The TC, as mentioned, is the local government responsible for setting the prices on the roads, as well as the fines for dishonest drivers who are caught. The TC is also responsible for performing spot checks to ensure that drivers are behaving honestly. Although this presents a new computational burden for the TC (as compared to PrETP, for example, which has the TSP performing the spot checks), we believe that it is important to keep all location information completely hidden from the TSP, as it is a business with incentive to sell this information. Since the TC already sees where each car is driving regardless of which body performs the spot checks (since it is the one operating the cameras), having it perform the audits itself minimizes the privacy lost by the driver.

Note, however, that the formal guarantees of correctness, security, and privacy provided by our system do not depend on having the TSP and TC not collaborate. In fact, both roles could be performed by a single organization. Since in practice businesses such as E-ZPass play the role of TSP, we recommend the separation of duties above to avoid giving the TSP an incentive to monetize customers' driving records. Of course, this assumes that regulation or the courts will forbid the government from misusing the information it collects.

2.2 Security model

In any privacy-preserving system, there are two goals which are absolutely essential to the success of the system: maintaining privacy, while still keeping users of the system honest. We discuss what this means in the context of electronic toll collection in the following two points:

- **Driver privacy**: Drivers should be able to keep their locations completely hidden from any other drivers who may want to intercept (and possibly modify) their payment information on its way to the TSP. With the exception of the random spot checks performed by the audit authority (in our case the TC), the locations of the driver should also be kept private from both the TC and the TSP. This property should hold even for a malicious TSP; as for the TC, we would like to guarantee that, as a result of the audit protocol, it learns only whether the driver was present at certain locations and times of its choice, even if it is malicious. The number of these locations and times about which the TC can query is fixed and a parameter of the audit protocol. An honest-but-curious TC will query the driver at those locations and times where she was actually observed, but a malicious TC might query for locations where no camera was present; see Section 4.3 for further discussion.

- **Driver honesty**: Drivers should not be able to tamper with the OBU to produce incorrect location or price information; i.e., pretending they were in a given location, using lower prices than are actually assigned, or simply turning off the OBU to pretend they drove less than they actually did. This property should hold even if drivers are colluding with other dishonest drivers, *and should in fact hold even if every driver in the system is dishonest.*

These security goals should look fairly similar to those outlined in previous work (e.g., PrETP or VPriv [38], and inspired by the earlier work of Blumberg, Keeler, and shelat [8]), but we note the consideration of possibly colluding drivers as an essential addition. We also note that we do not consider physical attacks (i.e., a malicious party gaining physical access to a driver's car) in this model, as we believe these attacks to be out of scope.

For ideal privacy, the locations of each driver would be kept entirely private even from the TC. This does not seem to be possible, however, as it would allow drivers to behave dishonestly without any risk of getting caught. Because each camera does take away some degree of privacy from the driver, we would like to minimize the number of cameras operated by the TC; at the same time, we need to keep it high enough so that the TC will have a very good chance of catching any cheating drivers. We believe this to be a fundamental limitation on the value of any privacy-preserving tolling system, however, as they are privacy preserving only when the spot-check cameras do not monitor such a large fraction of trips that the records themselves constitute a substantial privacy violation. As Blumberg, Keeler, and shelat write, "Extensive camera networks are simply not compatible with the kinds of privacy we demand since they collect too much information. If misused, they can provide adequate data for real-time tracking of vehicles" [8].

Finally, we note that these security properties are both achieved by Milo, under the assumption that cameras are randomly placed and invisible to drivers (i.e., the only way camera locations can leak to drivers is during the audit protocol). We discuss the potential issues with this assumption in Section 6.

3 Cryptographic Background

Because our scheme follows closely the PrETP construction [4], we employ the same modern cryptographic primitives as they do: commitment schemes and zero-knowledge proofs, in addition to the more familiar primitive of digital signatures [26]. In addition, to keep the spot-check camera locations hidden from drivers, we make use of another primitive, *blind identity-based encryption*, in a manner that is inspired by the oblivious transfer protocol of Green and Hohenberger [28].

3.1 Commitments

A commitment scheme is essentially the cryptographic relative of an envelope, and consists of two main phases: forming the commitment and opening the commitment. First, to form a commitment to a certain value, a user Alice can put the value in the envelope and then seal the envelope; to keep the analogy going, let's also assume she sealed it in some special way such that only she can open it. The sealed envelope then acts as her commitment, which she can send on to another user Bob. When the time comes, Alice can reveal the committed value by opening the envelope and showing Bob its contents. There are two properties that commitment schemes satisfy: *hiding* and *binding*. The hiding property says that, because Alice is the only one who can unseal the envelope, Bob will not be able to learn any information about its contents before she reveals them. In addition, the binding property says that, because the envelope is sealed, Alice will not be able to open it, change the value inside, and give it back to Bob without him noticing. In other words, when Alice finally reveals the opening of the commitment, Bob can be satisfied that those were the values inside all along. We will use the notation $c = \mathsf{Com}(m;r)$ to mean that c is a commitment to the message m using some randomness r. (Note that there are also some parameters involved, but that here and in the primitives that follow we omit them for simplicity.)

One more property we will require of our commitment schemes is that they are *additively homomorphic*. This means that there is an operation on commitments, call it $\boxplus$, such that if c_1 is a commitment to m_1 and c_2 is a commitment to m_2, then $c_1 \boxplus c_2$ will be a commitment to $m_1 + m_2$. This property can be achieved by a variety of schemes; to best suit our purposes, we work with Fujisaki-Okamoto commitments [18, 22], which rely on the Strong RSA assumption for their security.

3.2 Zero-knowledge proofs

Our second primitive, zero-knowledge proofs [24, 25], provides a way for someone to prove to someone else that a certain statement is true without revealing anything beyond the validity of the statement. For example, a user of a protected system might want to prove the statement "I have the password corresponding to this username" without revealing the password itself. The two main properties of zero-knowledge proofs are *soundness* and *zero knowledge*. Soundness guarantees that the verifier will not accept a proof for a statement that is false; in the above example, this means that the system will accept the proof only if the prover really does have the password. Zero knowledge, on the other hand, protects the prover's privacy and guarantees that the system in our example will not learn any information about the password itself, but only that the statement is true. A non-interactive zero-knowledge proof (NIZK for short) is a particularly desirable type of proof because, as the name indicates, it does not require any interaction between the prover and the verifier. For a given statement S, we will use the notation $\pi = \mathsf{NIZKProve}(S)$ to mean a NIZK formed by the prover for the statement S. Similarly, we will use $\mathsf{NIZKVerify}(\pi, S)$ to mean the process run by the verifier to check, using π, that S is in fact true. In our system we will need to prove only one type of statement, often called a *range proof*, which proves that a secret value x satisfies the inequality $lo \leq x < hi$, where lo and hi are both public. For this we can use Boudot range proofs and their extensions [11, 34], which are secure in the random oracle model [6] and assuming the Strong RSA assumption.

3.3 Blind identity-based encryption

Finally, to maintain driver honesty even in the case of possible collusions between drivers (as discussed in Section 2), we use an additional cryptographic primitive: *identity-based encryption* [10, 41]. Intuitively, identity-based encryption (IBE for short) extends the notion of standard public-key encryption by allowing a user's public key to be, rather than just a random collection of bits, some meaningful information relevant to their identity; e.g., their e-mail address. This is achieved through the use of an authority, who possesses some master secret key *msk* and can use it to provide secret keys corresponding to given identities on request (provided, of course, that the request comes from the right person). When we work with IBE, we will use the syntax $C = \mathsf{IBEnc}(id; m)$ to mean an identity-based encryption of the message m, intended for the person specified in the identity id. We will similarly use $m = \mathsf{IBDec}(sk_{id}; C)$ to mean the decryption of C using the secret key for the identity id.

Because of how IBE is integrated into our system, we will need the IBE to be augmented by a *blind* extraction protocol: a protocol interaction between a user and the authority at the end of which the user obtains the secret key corresponding to some identity of her choice, but the authority does not learn which identity was requested (and also does not learn the secret key for that identity). This process of getting the secret key will be denoted as $sk_{id} = \mathsf{BlindExtract}(id)$, keeping in mind that the authority learns neither id nor sk_{id}. As we show in Section 4, this property (introduced by Green and Hohenberger [28]) is crucial for guaranteeing that drivers do not learn where the TC has its cameras.

Furthermore, we would like our IBE to be *anonymous* [2], meaning that given a ciphertext C, a user cannot tell which identity the ciphertext is meant for (so, in particular, they cannot check to see if a guess is correct). Again, as we show in Section 4, this property is necessary to ensure that the TSP cannot simply guess and check where the driver was at a given time, and thus potentially learn information about her whereabouts.

To the best of our knowledge, there are two blind and anonymous IBEs in the cryptographic literature: the first due to Camenisch, Kohlweiss, Rial, and Sheedy [13] and the second to Green [27]; both are blind variants on the Boyen-Waters anonymous IBE [12]. While either of these schemes would certainly work for our purposes, we chose to come up with our own scheme in order to maximize efficiency. Our starting point is the Boneh-Franklin IBE [10], which is already anonymous [2, Section 4.5]. We then introduce a blind key-extraction protocol for Boneh-Franklin, based on the Boldyreva blind signature [9]. Finally, we "twin" the entire scheme to essentially run two copies in parallel; this is just to facilitate a "Twin Diffie-Hellman" style security proof [15]. We give a full description of our scheme in Appendix A, as well as a proof of its security in a variant of the Green-Hohenberger security model. Our IBE is conveniently efficient, but we stress that the Milo system could be instantiated with any provably secure IBE that is both blind and anonymous (and in particular the schemes of Camenisch et al. and Green which, while not as efficient as our scheme, have the attractive properties that they use significantly weaker assumptions and do not rely on random oracles in their proofs of security).

In the broadest sense, our blind IBE can be viewed as a special case of a secure two-party computation between the OBU and the TC, at the end of which the TC learns whether or not the driver paid honestly for a given segment, and the driver learns nothing (and in particular does not learn which segment the TC saw her in). As such, any efficient instantiation of this protocol as a secure two-party computation would be sufficient for our purposes. One promising approach, suggested by an anoymous reviewer, uses an *oblivious pseudorandom function* (OPRF for short) as a building block. With an OPRF, a user with access to a seed k for a PRF f and another user with input x can securely evaluate $f_k(x)$ without the first user's learning x or $f_k(x)$, and without the second user's learning the seed k; this can be directly applied to our setting by treating the seed k as a value known by the OBU, and the input x as the segment in which the TC saw the driver. An efficient OPRF was recently given by Jarecki and Liu [32]. Compared to our approach, the OPRFs of Jarecki and Liu may require increased interaction (which has implications for concurrent security) and potentially more computation than ours.

4 Our Construction

In this section, we describe the various protocols used within our system and how they meet the security goals described in Section 2.2; we note that only Algorithm 4.3 substantially differs from what is used in PrETP. There are three main phases we consider: the initialization of the OBU, the forming and verifying of the payments performed by the OBU and the TSP respectively, and the audit between the TC and the OBU. Below, we will detail the functioning of each of these algorithms; first, though, we give some intuition for how our scheme works and why the use of blind identity-based encryption means the audit protocol does not leak the locations of spot-check cameras to drivers.

In the audit protocol, the driver needs to show that her actual driving is consistent with the fee she chose to pay. To do this, she must upload her (claimed) driving history to the TSP's server; if she didn't, the TSP would have nothing to check the correctness of. Obviously, simply

uploading this history in the clear would provide no privacy. The VPriv system sidesteps this by having the driver upload the segments anonymously (using an anonymizing service such as Tor [20]), accompanied by a "tag" that will allow her to claim them as her own. We instead follow PrETP in having the driver upload a commitment of sorts to each of her segments. In addition, the driver commits to the cost associated with each segment using the additively homomorphic commitment scheme. Checking that the total payment is the sum of the fees for each committed segment is now easy: using the homomorphic operation $\boxplus$, the TSP can compute a commitment to the sum of the committed fees; the driver then provides the opening of this sum commitment, showing that its value is the fee she paid.[3]

What remains is to prove that the committed segments the driver uploaded to the server are in fact the segments she drove, and that the committed fee she uploaded alongside each is in fact the fee charged for driving it. Following VPriv, PrETP, and de Jonge and Jacobs' system (see Section 7), we rely on spot check cameras. The TC's cameras observed the driver at a few locations over the course of the month. It now challenges the driver to show that these locations are among the committed segments, and that the corresponding committed fees are correct. If the driver cannot show a commitment that opens to one of these spot check locations, she has been caught cheating; if the spot check locations are unpredictable then a simple probability analysis (see Section 6.1) shows that a cheating driver will likely be caught. In PrETP, the spot check has the TC sending to the driver the locations and times where she was observed; the driver returns the index and opening of the corresponding committed segments. This, of course, leaks the spot check locations to the driver. To get around this, we must somehow transmit the appropriate openings to the TC without the driver finding out which commitments are being opened.

Identity-based encryption allows us to achieve exactly the requirement above. Along with each of her commitments, the driver encrypts the opening of the commitment using IBE; the identity to which a commitment is encrypted is the segment location and time. She sends these encrypted openings to the TC along with the commitments themselves. (Note that it is crucial the ciphertext not reveal the identity to which it was encrypted, since otherwise the TSP and TC would learn the driver's entire driving history. This is why we require an anonymous IBE.) Now, if the TC had the secret key for the identity corresponding to the place and time where the driver was spotted, it could decrypt the appropriate ciphertext, obtain the commitment opening value, and check that the corresponding commitment was properly formed. But the TC can't ask the driver for the secret key, since this would also leak the spot-check location. Instead, it engages with the driver in a blind key-extraction protocol. The TC provides as input the location and time of the spot check and obtains the corresponding secret key without the driver learning which identity (i.e., location and time) was requested. By undertaking the blind extraction protocol only a certain number of times, the driver limits the number of spot checks the TC can perform.

Note that this is essentially an oblivious transfer protocol; our solution is in fact closely related to the oblivious transfer protocol of Green and Hohenberger [28], who introduced blind IBE.

Before any of the three phases can take place, the TC first decides on the segments used for payment and how much each one actually costs. It starts by dividing each road into segments of some appropriate length, for example one city block in denser urban areas or one mile along a highway in less congested areas. Because prices might change according to time of day, the TC also decides on a division of time into discrete quanta based on some "time step" when a new segment must be recorded by the OBU (even if the location endpoint has not yet been reached). For example, if two location endpoints are set as Exit 17 and Exit 18 on a highway and the time step is set to be a minute, then a driver traveling between these exits for more than a minute will have segments with the same location endpoints, but different time endpoints. In particular, if this driver starts at 22:00 and takes about three minutes to get from one exit to the other, she will end up with three segments:[4]

- $\big((\text{exit } 17, \text{exit } 18), (22{:}00, 22{:}01)\big)$;
- $\big((\text{exit } 17, \text{exit } 18), (22{:}01, 22{:}02)\big)$; and
- $\big((\text{exit } 17, \text{exit } 18), (22{:}02, 22{:}03)\big)$.

Each segment is of the form $\big((loc_1, loc_2), (time_1, time_2)\big)$; in the future, we denote these segments as $(where, when)$, where *where* represents the physical limits of the segment and *when* represents the particular time quantum during which the driver was in the segment *where*. For each of these segments, the TC will have assigned some price; this can be thought of as a publicly available function $f : (where, when) \to [0, M]$, where M is the maximum price assigned by the TC.

4.1 Initialization

Before any payments can be made, there are a number of parameters that need to be loaded onto the OBU. To start,

[3] There is a technicality here: range proofs are needed to prevent the driver from artificially reducing the amount she owes by committing to a few negative values. See Section 4.2 for more on this.

[4] In practice, the segment information will of course be more detailed; as a byproduct of using GPS anyway, each car will have access to precise coordinate and time information (including date).

the OBU will be given some unique value to identify itself to the TSP; we refer to this value as *tag*. Because the OBU will be signing all the messages it sends, it first needs to generate a signing keypair (vk_{tag}, sk_{tag}); the public verification key will need to be stored with both the TSP and TC, while the signing key will be kept private. We will also use an augmented version of the BlindExtract protocol (mentioned in Section 3.3) in which the OBU and TC will sign their messages to each other, which means the OBU will need to have the verification key for the TC stored as well (alternatively, they could just communicate using a secure channel such as TLS). In addition, the OBU will need to generate parameters for an IBE scheme in which it possesses the master secret key *msk*, as well as to load the parameters for the commitment and NIZK schemes (note that it is important the OBU does not generate these latter parameters itself, as otherwise the driver would be able to cheat). Finally, the OBU will also need to have stored the function f used to define road prices.

4.2 Payments

Once the OBU is set up with all the necessary parameters, it can begin making payments. As the driver travels, the GPS picks up location and time information, which can then be matched to segments $(where, when)$. For each of these segments, the OBU first computes the cost for that segment as $p = f(where, when)$. It then computes a commitment c to this value p; we will refer to the opening of this commitment as $open_c$. Next, the OBU computes an identity-based encryption C of the opening $open_c$ along with a confirmation value 0^λ, using the identity $id = (where, when)$. Finally, the OBU computes a non-interactive zero-knowledge proof π that the value contained in c is in the range $[0, M]$. This process is then repeated for every segment driven, so that by the end of the month the OBU will end up with a set of tuples $\{(c_i, C_i, \pi_i)\}_{i=1}^n$. In addition to this set, the OBU will also need to compute the opening $open_{final}$ for the commitment $c_{final} = c_1 \boxplus c_2 \boxplus \cdots \boxplus c_n$; i.e., the opening for the commitment to the sum of the prices, which effectively reveals how much the driver owes. The OBU then creates the final message $m = (tag, open_{final}, \{(c_i, C_i, \pi_i)\}_i)$, signs it to get a signature σ_m, and sends to the TSP the tuple (m, σ_m). This payment process is summarized in Algorithm 4.1. The parameter λ, set to 160 for 80-bit security, is explained below.

Once the TSP has received this tuple, it first looks up the verification key for the signature using *tag*. If it is satisfied that this message came from the right OBU, then it performs several checks; if not, it aborts and alerts the OBU that something went wrong (i.e., the message was manipulated in transit) and it should resend the tuple. Next, it checks that each commitment c_i was properly

Algorithm 4.1: Pay, run by the OBU

Input: segments $\{(where_i, when_i)\}_{i=1}^n$, identifier *tag*, signing key sk_{tag}

forall $1 \le i \le n$ **do**
  $p_i = f(where_i, when_i)$
  $c_i = \mathsf{Com}(p_i; r_i)$
  $C_i = \mathsf{IBEnc}((where_i, when_i); (p_i; r_i; 0^\lambda))$
  $\pi_i = \mathsf{NIZKProve}(0 \le p_i \le M)$
$open_{final} = (\sum_i p_i; \sum_i r_i)$
$m = (tag, open_{final}, \{(c_i, C_i, \pi_i)\}_{i=1}^n)$
$\sigma_m = \mathsf{Sign}(sk_{tag}, m)$
return (m, σ_m)

formed by acting as the verifier for the NIZK π_i; if one of these checks failed then it knows that the driver committed to an incorrect price (for example, a negative price to try to drive down her monthly bill). The TSP then performs the homomorphic operation on the commitments to get $c_{final} = c_1 \boxplus c_2 \boxplus \cdots \boxplus c_n$ and checks that $open_{final}$ is the opening for c_{final}. If all these checks pass, the TSP can debit p_{final} (contained in $open_{final}$) from the user's account; if not, something has gone wrong and the TSP can flag the driver as suspicious and continue on to legal proceedings, as is done with current traffic violations. This algorithm is summarized in Algorithm 4.2.

In terms of privacy, the hiding property of the commitment scheme and the zero knowledge property of the NIZK scheme guarantee that the driver's information is being kept private from the TSP. Furthermore, the anonymity of the IBE scheme guarantees that, although the segments are used as the identity for the ciphertexts C_i, the TSP will be unable to learn this information given just these ciphertexts. In addition, some degree of honesty is guaranteed. First, because the message was signed by the OBU, the TSP can be sure that the tuple came from the correct driver and not some other malicious driver trying to pass herself off as someone else (or cause the first driver to pay more than she owes). Furthermore, if all the checks pass then the binding property of the commitment scheme and the soundness property of the NIZK scheme guarantee that the values contained in the commitments are to valid prices and so the TSP can be somewhat convinced that the price p_{final} given by the driver is the correct price she owes for the month. The TSP cannot, however, be convinced yet that the driver did not simply turn off her OBU or otherwise fake location or price information; for this, it will need to forward the payment tuple to the TC, which initiates the audit phase of the protocol.

4.3 Auditing

As we argued in Section 2.2, although the audit protocol does take away some degree of privacy from the driver, this small privacy loss is necessary to ensure honesty

Algorithm 4.2: VerifyPayment, run by the TSP

Input: payment tuple (m, σ_m), verification key vk_{tag}

if $\mathsf{SigVerify}(vk_{tag}, m, \sigma_m) = 0$ **then**
  return $\perp$
parse m as $\left(tag, open_{final}, \left\{(c_i, C_i, \pi_i)\right\}_{i=1}^{n}\right)$
forall $1 \leq i \leq n$ **do**
  if $\mathsf{NIZKVerify}\left((c_i, \pi_i), 0 \leq p_i \leq M\right) = 0$ **then**
    return suspicious
$c_{final} = c_1 \boxplus \cdots \boxplus c_n$
if $c_{final} = \mathsf{Com}(open_{final})$ **then**
  parse $open_{final}$ as $(p_{final}; r_{final})$
  debit account for tag by p_{final}
  return okay
else
  return suspicious

within the system. We additionally argued that the TC should not reveal to the driver the locations of the cameras and furthermore believe that the driver should not even learn the number of cameras at which the TC saw her, as even this information would give her opportunity to cheat (for more on this see Section 6). We therefore assume that the TC makes some fixed number of queries k for every driver, regardless of whether or not it has in fact seen the driver k times. To satisfy this assumption, if the TC has seen the driver on more than k cameras, it will just pick the first k (or pick k at random, it doesn't matter) and query on those. If it has seen the driver on fewer than k cameras, we can designate some segment to be a "dummy" segment, which essentially does not correspond to any real location/time tuple. The TC can then query on this dummy segment until it has made k queries in total; because the part of the protocol in which the TC performs its queries is blind, the OBU won't know that it is being queried on the same segment multiple times.

After the TSP has forwarded the OBU's payment tuple to the TC, the TC first checks that the message really came from the OBU (and not, for example, from a malicious user or even the TSP trying to frame the driver). As with the TSP, if this check fails then it can abort the protocol and alert the OBU or TSP. It then extracts the tuples $\left\{(c_i, C_i, \pi_i)\right\}$ from m and begins issuing its random spot checks to ensure that the driver was not lying about her whereabouts. This process is outlined in Algorithm 4.3. Because there were a certain number of cameras the driver passed, the TC will have a set of tuples $\left\{(loc_i, time_i)\right\}$ of its own that correspond to the places and times at which the TC saw the driver. First, for every pair $(loc, time)$, the TC will need to determine which segment this pair belongs to; this then gives it a set $\left\{(where_i, when_i)\right\}$ of tuples that the driver would have logged if they were behaving honestly (unless the set has been augmented by the dummy segment as described above, in which case the OBU clearly will not have logged this segment).

After the TC has this set of tuples, it uses the identity-based encryption C_j contained within every tuple sent by the OBU. Recall from Algorithm 4.1 that the identity corresponding to each encryption is the segment $(where_j, when_j)$, and that the encryption itself is of the opening of the commitment c_j (contained in the same tuple), along with a confirmation value 0^λ. Therefore, if the TC can obtain the secret key sk_{id} from the OBU for the identity $id = (where_j, when_j)$, then it can successfully decrypt the ciphertext and obtain the opening for the commitment, which it can then use to check if the driver is recording correct price information. Because the TC does not know which ciphertext corresponds to which segment, however, once the TC obtains this secret key it will then need to attempt to decrypt each C_j.

To prevent drivers from using a single commitment to pay for two segments, we require that it be computationally difficult to find a ciphertext C that has valid decryptions under two identities id_1 and id_2. For our IBE, it is sufficient to encrypt a confirmation value 0^λ along with the message (where $\lambda = 160$ for 80-bit security), since messages are blinded with a random oracle hash that takes the identity as input. On decryption, one checks that the correct confirmation value is present. Note that we do not require CCA security.

If C_j does decrypt properly for some j, then the TC checks that the value contained inside is the opening of the commitment c_j. If it is, then the TC further checks that the price p_j is the correct price for that road segment by computing $f(where_j, when_j)$. If this holds as well, then the TC can be satisfied that the driver paid correctly for the segment of the road on which she was seen and move on to the next camera. If it does not hold, then the TC has reason to believe that the driver lied about the price of the road she was driving on. If instead the opening is not valid, the TC has reason to believe that the driver formed either the ciphertext C_j or the commitment c_j incorrectly. Finally, if none of the ciphertexts properly decrypted using sk_{id} (i.e., C_j did not decrypt for any value of j), then the TC knows that the driver simply omitted the segment $(where_j, when_j)$ from her payment in an attempt to pretend she drove less. In any of these cases, the TC believes the driver was cheating in some way and can undertake legal proceedings. If all of these checks pass for every camera, then the driver has successfully passed the audit and the TC is free to move on to another user.

In terms of driver honesty, the addition of BlindExtract allows the TC to obtain sk_{id} without the OBU learning the identity, and thus the location at which they were caught on camera. As argued in Section 2, this is absolutely cru-

Algorithm 4.3: Audit, run by the TC

Input: payment tuple (m, σ_m), camera tuples $\{(loc_i, time_i)\}_{i=1}^{k}$, verification key vk_{tag}

1 **if** $\mathsf{SigVerify}(vk_{tag}, m, \sigma_m) = 0$ **then**
2 **return** $\perp$
3 parse m as $(tag, open_{final}, \{(c_j, C_j, \pi_j)\}_{j=1}^{n})$
4 **forall** $1 \leq i \leq k$ **do**
5 determine segment $(where_i, when_i)$ for $(loc_i, time_i)$
6 $sk_i = \mathsf{BlindExtract}(where_i, when_i)$
7 match = 0
8 **forall** $1 \leq j \leq n$ **do**
9 $m_j = \mathsf{IBDec}(sk_i; C_j)$
10 **if** m_j *parses as* $(p_j; r_j; 0^{\lambda})$ **then**
11 match = 1
12 **if** $\mathsf{Com}(m_j) \neq c_j$ **then**
13 **return** suspicious
14 **if** $p_j \neq f(where_i, when_i)$ **then**
15 **return** suspicious
16 **break**
17 **if** *match = 0* **then**
18 **return** suspicious
19 **return** okay

cial for maintaining driver honesty, both individually and in the face of possible collusions. In terms of privacy, if the OBU and TC sign their messages in the BlindExtract phase, then we can guarantee that no malicious third party can alter messages in their interaction in an attempt to learn the segment in which the driver was caught on camera (or, alternatively, frame the driver by corrupting sk_{id}). As mentioned in Section 2, whereas the cameras do take away some part of the driver's privacy, they are necessary to maintain honesty; we also note that no additional information is revealed throughout the course of this audit interaction provided both parties behave honestly. One potential downside of this protocol, however, is that the TC is not restricted to querying locations at which it had cameras; it can essentially query any location it wants without the driver's knowledge (although the driver is at least aware of how many queries are being made). We believe that our system could be augmented to resist such misbehavior through an "audit protocol audit protocol" that requires the TC to demonstrate that it actually has camera records corresponding to some small fraction of the spot check it performs, much as its own audit protocol requires the driver to reveal some small fraction of its segments driven. This "audit audit" could be performed on behalf of drivers by an organization such as EFF or the ACLU; alternatively, in some legal settings an exclusionary rule could be introduced that invalidates evidence obtained through auditing authority misbehavior.

	Time (ms)	
Operation	Laptop	ARM
Creating parameters	75.12	1083.61
Encryption	82.11	1187.82
Blind extraction (user)	13.13	214.06
Blind extraction (authority)	11.21	175.25
Decryption	78.31	1131.58

Table 1: The average time, in milliseconds and over a run of 10, for the various operations in our blind IBE protocol, performed on both a MacBook Pro and an ARM v5TE. The numbers for encryption and decryption represent the time taken to encrypt/decrypt a pair of 1024-bit numbers using the curve $y^2 = x^3 + x \bmod p$ at the 80-bit security level, and the numbers for blind extraction represent the time to complete the computation required for each side of the interactive protocol.

5 Implementation and Performance

In order to achieve a more effective audit protocol, an extra computational burden is required for both the OBU and the TC. In this section, we consider just how great this additional burden is; in particular, we focus on our blind identity-based encryption protocol from Appendix A, as well as Algorithm 4.3 from Section 4.3. The benchmarks presented for these protocols were collected on two machines: a MacBook Pro running Mac OS X 10.6 with a 2.53 GHz Intel Core 2 Duo processor and 4 GB of RAM, and an ARM v5TE running Linux 2.6.24 with a 520 MHz processor and 128 MB of RAM. We believe that the former represents a fairly conservative estimate for the amount of computational resources available to the TC, whereas the latter represents a machine that could potentially be used as an OBU. For the bilinear groups needed for blind IBE we used the supersingular curve $y^2 = x^3 + x \bmod p$ for a large prime p (which has embedding degree 2) within version 5.4.3 of the MIRACL library [40], and for the NIZKs and commitments we used ZKPDL (Zero-Knowledge Proof Description Language) [35], which itself uses the GNU multi-precision library [23] for modular arithmetic.

Table 1 shows the time taken for each of the unit operations performed within the IBE scheme. As mentioned in Section 4, in the context of our system the creation of the parameters will be performed when the OBU is initialized, the encryption will be performed during the Pay protocol (line 4 of Algorithm 4.1), and both blind extraction and decryption will be performed in the audit phase between the TC and the OBU (lines 6 and 9 of Algorithm 4.3 respectively).

We consider the computational costs for the OBU and the TC separately, as well as the communication overhead for the whole system.[5]

[5] We do not consider the computational costs for the TSP here, as

OBU computational costs. During the course of a month (or however long an audit period is), the OBU is required to spend time performing computations for two distinct phases of the Milo protocol. The first phase is the Pay protocol, which consists of computing the commitments to segment prices, encrypting the openings of the commitments, and producing a zero-knowledge proof that the value in the commitment lies in the right range. From Table 1, we know that encryption takes roughly a second when encrypting 1024-bit number on the ARM. As these correspond to "medium security" in PrETP [4, Table 2], and our commitments and zero-knowledge proofs are essentially identical to theirs, we can use the relevant timings from PrETP to see that the total time taken for the Pay protocol should be at most 20 seconds per segment. As long as the time steps are at least 20 seconds and the segment lengths are at least half a mile (assuming people drive at most 90 miles per hour), the calculations can therefore be done in real time.

The second phase of computation is the end of the month audit protocol. Here, the OBU is responsible for acting as the IBE authority to answer blind extraction queries from the TC. As we can see in Table 1, each query takes the OBU approximately 175 milliseconds, independent of the number of segments. If the TC makes a small, fixed number of queries, say ten, for each vehicle, then the OBU will spend only a few seconds in the Audit protocol each month.

TC computational costs. In the course of the Audit protocol, the TC has to perform a number of complex calculations. In particular, the cost of challenging the OBU for each camera is proportional to the number of segments the OBU reported driving.

To obtain our performance numbers for the audit protocol, we considered the driving habits of an average American, both in terms of time spent and distance driven. For time, we assumed that an average user would have a commute of 30 minutes each way, meaning one hour each day, in addition to driving between two and three hours each weekend. For distance, we assumed that an average user would drive around 1,000 miles each month. While we realize that these averages will vary greatly between locations (for example, between a city and a rural area), we believe that these measures still give us a relatively realistic setting in which to consider our system.

Table 2 gives the time it takes for the TC to challenge the OBU on a single segment for several segment lengths and time steps; we can see that the time taken grows approximately linearly with the number of segments. To determine the number of segments, we considered both fine-grained and coarse-grained approaches. For the fine-grained approach, we considered a time step of one minute. Using our assumptions about driving habits, this means that in a 30-day month with 22 weekdays, our average user will drive approximately 1,320 segments. Adding on an extra 680 segments for weekends, we can see that a user might accumulate up to 2,000 segments in a month. In the way that road prices are currently decided, however, a time step of one minute seems overly short, as typically there are only two times considered throughout the day: peak and off-peak. We therefore considered next a time step of one hour, keeping our segment length at 1 mile. Here the number of miles driven determines the number of segments rather than the minutes spent in the car, and so we end up with approximately 1,000 segments for the month. Finally, we considered a segment length of 2 miles, keeping our time step at one hour; we can see that this results in approximately half as many segments as before, around 500 segments. Longer average physical segment lengths would result in an even lower number of segments (and therefore better performance).

Communication overhead. Looking at Table 3, we can see that the size of a payment message is approximately 6kB per segment; furthermore, this size is dominated by the NIZK (recall that each segment requires a commitment, a NIZK, and a ciphertext), which accounts for over 90% of the total size. For our parameter choices in Table 2, this would result in a total payment size of approximately 11MB in the worst case (with 2000 segments) and 3MB in the best case (with 500 segments). In PrETP, on the other hand, the authors claim to have sizes of only 1.5kB per segment [4, Section 4.3]. Using their more compact segments with our ciphertexts added on would therefore result in a segment size of only 2kB, which means the worst-case size of the entire payment message would be under 4MB (and the best-case size approximately 1MB).

Finally, we can see that the overhead for the rest of the Audit protocol is quite small: each blind IBE key sent from the OBU to the TC is only 494 bytes; if the TC makes ten queries per audit, then the total data transferred in the course of the protocol is about 5kB.

5.1 Milo cost analysis

If we continue to assume that the TC always queries the user on ten cameras, then the entire auditing process will take less than 10 minutes per user in the worst case (when there are 2,000 segments) and less than 2 minutes in the best case (when there are 500 segments). If we consider pricing computational resources as according to Amazon EC2 [3], then to approximately match the computer used for our benchmarks would cost about 10 cents per hour. Between 6 and 30 users can be audited within an hour, so

they are essentially the same as they were within PrETP; the numbers they provide should therefore provide a reasonably accurate estimate for the cost of the TSP within our system as well.

Length	Time step	Segments	Time for TC (s)
1 mile	1 minute	2000	55.68
1 mile	1 hour	1000	33.51
2 miles	1 hour	500	10.45

Table 2: The average time, in seconds and over a run of 10, for the TC to perform a single spot check given segment lengths and time steps; we consider only the active time spent and not the time waiting for the OBU. Essentially all of the time was spent iterating over the segments; as such, the time taken grows approximately linearly with the number of segments. To determine the approximate number of segments given segment lengths and time steps, we assumed that an average user would drive for 1,000 miles in a 30-day month, or about 33 hours (1 hour each weekday and an extra 11 hours over four weekends).

Object	Size (B)
NIZK	5455
Commitment	130
Ciphertext	366
Total Pay segment	5955
Audit message	494

Table 3: Size of each of the components that needs to be sent between the OBU and the TC, in bytes. Each segment of the payment consists of a NIZK, commitment, and ciphertext; all the segments are forwarded to the TC from the TSP at the start of an audit. In the course of the Audit protocol the OBU must also send blind IBE keys to the TC.

each user ends up costing the system between one-third of a cent and 2 cents each month; this is an amount that the TSP could easily charge the users if need be (although the cost would presumably be cheaper if the TC simply performed the computations itself). We therefore believe that the amount of computation required to perform the audits, in addition to being necessary in guaranteeing fairness and honesty within the system, is reasonably practical.

Finally, to examine how much Milo would cost if deployed in a real population we consider the county of San Diego, which consists of 3 million people possessing approximately 1.8 million vehicles, and almost 2,800 miles of roads [16, 17, 43]. As we just saw, Milo has a computational cost of up to 2 cents per user per month, which means a worst-case expected annual cost of $432,000; in the best case, wherein users cost only one-third of a cent per month, the expected annual cost is only $72,000. In the next section, we can see how these costs compares to that of the "naïve" solution to collusion protection; i.e., one in which we attempt to protect against driver collusion through placement of cameras as opposed to prevention and protection at the system level.

6 Collusion Resistance

Previously proposed tolling systems did not take collusion into account, as they allow the auditing authority to transmit camera locations in the clear to drivers. Given these locations, colluding drivers can then share their audit transcripts each month in order to learn a greater number of camera locations than they would have learned alone. Furthermore, websites already exist which record locations of red light cameras [37] and speed cameras [36]; one can easily imagine websites similar to these that collect crowd-based reports of audit camera locations. With cameras whose locations are fixed from month to month, the cost to cheat is therefore essentially zero (just check the website!) and so we can and should expect enterprising drivers to take advantage of the system. In contrast, Milo is specifically designed to prevent these sorts of trivial collusion attacks.

In addition to learning camera locations through the course of the audit phase, drivers may also learn camera locations from simply seeing them on the road. This is also quite damaging to the system, as drivers can learn the locations of cameras simply by spotting them. After pooling together the various locations and times at which they saw cameras, cheating drivers can fix up their driving record in time to pass any end-of-month audit protocol.

To prevent such cheating, a system could instead require the OBU to transmit the tuples corresponding to segments as they are driven, rather than all together at the end of the month. Without an anonymizing service such as Tor (used in VPriv [38]), transmitting data while driving represents too great a privacy loss, as the TSP can easily determine when and for how long each driver is using their car. One possible fix might seem to be to continually transmit dummy segments while the car is not in use; transmitting segments in real time over a cellular network, however, leaks coarse-grained real-time location information to nearby cell towers (for example, staying connected to a single tower for many hours suggests that you are stationary), thus defeating the main goal of preserving driver privacy.

Finally, we note that there exists a class of expensive physical attacks targeting any real-world implementation of a camera-based audit protocol. For example, against fixed-location cameras, cheating drivers could disable their OBU for specific segments each month, revealing information about those segments. Against mobile cameras, a driver could follow each audit vehicle and record its path, sharing with other cheating drivers as they go. One can imagine defenses against these attacks and even more

fanciful attacks in response; these sort of attacks quickly become very expensive and impractical, however, and provide tell-tale signs of collusion (e.g., repeated cheating, suspicious vehicles). We therefore do not provide a system-level defense against them.

6.1 Collusion resistance cost analysis

With Milo, we have modified the PrETP system to avoid leaking the locations of cameras as part of the audit protocol. An alternative approach is to leave PrETP (or one of the other previously proposed solutions) in place and increase the number of audit cameras and their mobility, thus reducing the useful information leaked in audits even when drivers collude. Whereas deploying Milo would increase computational costs over PrETP, deploying the second solution would increase the operational costs associated with running the mobile audit cameras. In this section, we compare the costs associated with the two solutions. Even with intentionally conservative estimates for the operating costs of mobile audit cameras, Milo appears to be competitive for reasonable parameter settings; as Moore's law makes computation less expensive, Milo will become more attractive still.

Hardening previous tolling systems against trivial driver collusion is possible if we consider using continuously moving, invisible cameras. Intuitively, if cameras move randomly, then knowing the position and time at which one audit camera was seen does not allow other cheating drivers to predict any future camera locations. The easiest way to achieve these random spot checks is to mount cameras on special-purpose vehicles, which then perform a random walk across all streets in the audit area. Even this will not generate truly random checks (as cars must travel linearly through streets and obey traffic laws); for ease of analysis we assume it does. Furthermore, we will make the assumptions that the audit vehicles will never check the same segment simultaneously, operate 24 hours a day (every day), and are indistinguishable from other cars; tolling segments are 1 mile; and non-audit vehicles drive all road segments with equal probability. These assumptions are by no means realistic, but they present a stronger case for moving cameras and so we use them, keeping in mind that any more realistic deployment will have higher cost.

Using a probability analysis similar to that of VPriv [38, Section 8.4], we consider an area with M miles of road and C audit vehicles. If both audit vehicles and other drivers are driving all roads uniformly at random, then a driver will share a segment with an audit vehicle with probability $p = \frac{C}{M}$ with each mile driven. If the driver travels m miles in a tolling period, she will be seen at least once by an audit vehicle with a probability of

$$1-(1-p)^m = 1-\left(\frac{M-C}{M}\right)^m. \qquad (1)$$

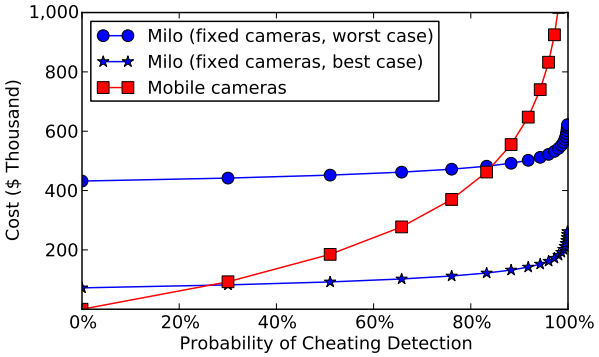

Figure 2: A cost comparison of using the Milo system against using mobile cameras within previously proposed systems. We know, from Section 5.1, that Milo has a worst-case computational cost of $432,000 per year and a best case of $72,000; for the other systems, we ignore computation completely (i.e., we assume it is free). Even with the minimal costs we have assigned to operating a fleet of audit vehicles 24 hours a day and assuming worst-case computational costs, Milo becomes equally cheap when the probability of catching cheating drivers is 83%, and becomes significantly cheaper as the probability approaches 100%. For Milo's best-case cost, it becomes cheaper as soon as more than one camera is used.

To determine the overall cost of this type of operation, we return to San Diego County (discussed already in Section 5.1); recall that it consists of 1.8 million vehicles driving on 2,800 miles of road, in which the average distance driven by one vehicle is 1,000 miles in a month. Using Equation 1, with one audit vehicle ($C = 1$), the probability that a driver gets caught is $1-(2799/2800)^{1000} \approx .3$, so that a potentially cheating driver still has a 70% chance of completely avoiding any audit vehicles for a month. If we use two audit vehicles, then this number drops to 49%. Continuing in this vein, we need 13 audit vehicles to guarantee a 99% chance of catching drivers who intentionally omit segments. Achieving these results requires the TC to employ drivers 24 hours a day, as well as purchase, maintain, and operate a fleet of audit vehicles. To consider the cost of doing so, we estimate the depreciation, maintenance, and operation cost of a single vehicle to be approximately $12,500 a year [44]. Furthermore, California has a minimum wage of $8.00/hr; paying this to operate a single vehicle results in minimum annual salary costs of $70,080, ignoring all overtime pay and benefits. Each audit vehicle will therefore cost at least $82,500 per year (ignoring a number of additional overhead costs).

Finally, we compare the cost of operating these mobile cameras with the cost of the Milo system. Because Milo leaks no information about camera locations to drivers, cameras can in fact stay at *fixed* locations; as long they are virtually invisible, drivers have no opportunities to learn their locations and so there is no need to move them

continuously. We therefore consider placing invisible cameras at random fixed locations, and can calculate the probability of drivers being caught by Milo using Equation 1, where we now use C to represent the number of cameras (and continue to assume that drivers drive 1,000 miles at random each month).

Figure 2 compares the cost of Milo with fixed cameras and the cost of previous systems with mobile cameras as the probability of detecting cheating increases. We used a per-camera annual cost of $10,000.[6] As we can see, in the worst case, Milo achieves cost parity with mobile cameras at a detection probability of 83% and becomes vastly cheaper as the systems approach complete coverage, while in the best case it achieves cost parity as soon as more than a single camera is used (which gives a detection probability of around 30%). With either of these numbers, we remember that our assumptions about the cost of operating these vehicles significantly underrated the actual cost; substituting in more realistic numbers would thus cause Milo to compare even more favorably. In addition, future developments in computing technology are almost guaranteed to drive down the cost of computation, while fuel and personnel costs are not likely to decrease, let alone as quickly. Therefore, we believe that Milo is and will continue to be an effective (and ultimately cost effective) solution to protect against driver collusion.

7 Related work

The study of privacy-preserving traffic enforcement and toll collection was initiated in papers by Blumberg, Keeler, and shelat [8] and Blumberg and Chase [7]. The former of these papers gave a system for traffic enforcement (such as red-light violations) and uses a private set-intersection protocol at its core; the latter gave a system for tolling and road pricing, and uses general secure function evaluation. Neither system keeps the location of enforcement or spot-check devices secret from drivers. In an important additional contribution, these papers formalized the "implicit privacy" that drivers currently enjoy: The police could tail particular cars to observe their whereabouts, but it would be impractical to apply such surveillance to more than a small fraction of all drivers.[7]

Another approach to privacy-preserving road pricing was given by Troncoso et al. [42], who proposed trusted tamper-resistant hardware in each car that calculates the required payment, and whose behavior can be audited by the car's owner. The Troncoso et al. paper also includes a useful survey of pay-as-you-drive systems deployed at the time of its publication. See Balasch, Verbauwhede, and Preneel [5] for a prototype implementation of the Troncoso et al. approach.

De Jonge and Jacobs [19] proposed a privacy-preserving tolling system in which drivers commit to the path they drove without revealing the individual road segments. De Jonge and Jacobs' system uses hash functions for commitments, making it very efficient. Only additive road pricing functions are allowed (i.e., ones for which the cost of driving along a path is the sum of the cost of driving along each segment of the path); this makes possible a protocol for verifying that the total fee was correctly computed as the sum of each road segment price by revealing, essentially, a path from the root to a single leaf in a Merkle hash tree. (This constitutes a small information leak.) In addition, de Jonge and Jacobs use spot checks to verify that the driver faithfully reported each road segment on which she drove.

More recently, Popa, Balakrishnan, and Blumberg proposed the VPriv [38] privacy-preserving toll collection system. VPriv takes advantage of the additive pricing functions it supports to enable the use of homomorphic commitments whereby the drivers commit to the prices for each segment of their path as well as the sum of the prices. Then, the product of the commitments is a commitment of the sum of the prices. This eliminates the need for a protocol to verify that the sum of segment prices was computed correctly. Like previous systems, VPriv uses (camera) spot checks to ensure that drivers faithfully reveal the segments they drove. The downside to VPriv is that, for the audit protocol, drivers must upload the road segments they drove to the server; to avoid linking these to their IP address, they must use an anonymizing network such as Tor.

Balasch et al. proposed PrETP [4] to address some of the shortcomings in VPriv. In PrETP, drivers do not reveal the road segments they drove in the clear, and so do not need an anonymizing network. Instead, they commit to the segments and, using a homomorphic commitment scheme, to the corresponding fees; in the audit protocol, they open the commitments corresponding to the road segments on which spot-check cameras observed them.

In each of the system of de Jonge and Jacobs, VPriv, and PrETP, drivers are challenged in the audit protocol to prove that they committed to or otherwise uploaded the segments for which there is photographic evidence

[6] This number was loosely choosen based upon purchase costs for red light violation cameras. Note that the choice does not affect the differential system cost, as both systems must operate the same number of cameras to achieve a given probability of success.

[7] We would like to correct one misconception, lest it influence future researchers. Blumberg, Keeler, and shelat write, "the standards of suspicion necessary to stop and search a vehicle are much more lax than those required to enter and search a private residence." In the U.S., the same standard — probable cause — governs searches of both vehicles and residences; the difference is only that a warrant is not required before the search of a car, as "it is not practicable to secure a warrant because the vehicle can be quickly moved out of the locality or jurisdiction in which the warrant must be sought" (*Carroll v. United States*, 267 U.S. 132 (1925), at 153).

that they were present. As discussed in Sections 1 and 2, this revealing of camera locations enables several attacks which allow drivers to pay less than their actual tolls. Additionally, camera placement and tolling areas must be restricted to ensure driver privacy, for example, by using "virtual trip lines" [30].

In recent work, Hoepman and Huitema [29] observed that in both VPriv and PrETP the audit protocol allows the government to query cars about locations where there was no camera, a capability that could be misused, for example, to identify whistleblowers. They propose a privacy-preserving tolling system in which vehicles can be spot-checked only where their presence was actually recorded, and in which overall driver privacy is guaranteed so long as the pricing provider and aggregation provider do not collaborate. Like VPriv, Hoepman and Huitema's system requires road segments to be transmitted from the car to the authority over an anonymizing network.

Besides tolling, there are other vehicular applications that require privacy guarantees; see, generally, Hubaux, Čapkun, and Luo [31]. One important application is vehicle-to-vehicle ad hoc safety networks [14]; see Freudiger et al. [21] for one approach to location privacy in such networks. Another important application is aggregate traffic data collection. Hoh et al. [30] propose "virtual trip lines" that instruct cars to transmit their location information and are placed to minimize privacy implications; Rass et al. [39] give an alternative construction based on cryptographic pseudonym systems.

Vehicle communication is one class of ubiquitous computing system. Location privacy in ubiquitous computing generally is a large and important research area; see the recent survey by Krumm [33] for references.

8 Conclusions

In recent years, privacy-preserving toll collection has been proposed as a way to implement more fine-grained pricing systems without having to sacrifice the privacy of drivers using the roads. In such systems drivers do not reveal their locations directly to the toll collection authorities; this means there needs to be a mechanism in place to guarantee that the drivers are still reporting their accumulated fees accurately and honestly. Maintaining this balance between privacy and honesty in an efficient and practical way has proved to be a challenging problem; previous work, however, such as the VPriv and PrETP systems, has demonstrated that this problem is in fact tractable. Both these systems employ modern cryptographic primitives to allow the driver to convince the collection authority of the accuracy of her payment without revealing any part of her driving history. To go along with this collection mechanism, a series of random spot checks (i.e., the authority challenging the driver to prove that she paid for segments in which she was caught on camera) must be performed in order to maintain honesty and fairness throughout the system.

In this paper, we have identified large-scale driver collusion as a realistic and damaging threat to the success of privacy-preserving toll collection systems. To protect against these sorts of collusions, we have presented Milo, a system which achieves the same privacy properties as VPriv and PrETP, but strengthens the guarantee of driver honesty by avoiding revealing camera locations to drivers. We have also implemented the new parts of our system to show that achieving this stronger security guarantee does not add an impractical burden to any party acting within the system.

Finally, along more practical lines, we have considered a naïve approach to protecting against collusions and shown that, from both a cost and effectiveness consideration, it is ultimately less desirable and more cumbersome than Milo.

The weaknesses we identify in previous systems are caused by the gap between the assumption made by the cryptographic protocols (that spot checks are unpredictable) and the real-world cameras used to implement them — cameras that are physical objects that can be identified and may be difficult to move. If drivers are able to avoid some cameras, more of them will be required; if too many spot-check cameras are deployed, the records they generate will themselves degrade driver privacy. We believe that it is important for work on privacy-preserving tolling to address this limitation by carefully considering how the spot checks it relies on will be implemented.

Acknowledgements

We gratefully acknowledge Matthew Green, our shepherd, as well as our anonymous reviewers for their valuable feedback. This material is based on work supported by the National Science Foundation under Grant No. CNS-0963702, and by the MURI program under AFOSR Grant No. FA9550-08-1-0352; the first author was also supported in part by a fellowship from the Powell Foundation.

References

[1] Recent Cases: D.C. Circuit deems warrantless use of GPS device an unreasonable search. *Harvard Law Review*, 124(3):827–34, Jan. 2011.

[2] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3), July 2008.

[3] Amazon Elastic Compute Cloud (EC2). Amazon EC2 pricing. http://aws.amazon.com/ec2/pricing.

[4] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens. PrETP: privacy-preserving toll pricing. In I. Goldberg, editor, *Proceedings of USENIX Security 2010*, pages 63–78. USENIX, Aug. 2010.

[5] J. Balasch, I. Verbauwhede, and B. Preneel. An embedded platform for privacy-friendly road charging applications. In W. Mueller, editor, *Proceedings of DATE 2010*, pages 867–872. IEEE Computer Society, Mar. 2010.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security (CCS) 1993*, pages 62–73, 1993.

[7] A. J. Blumberg and R. Chase. Congestion pricing that preserves 'driver privacy'. In U. Nunes, editor, *Proceedings of ITSC 2006*, pages 725–732. IEEE Intelligent Transportation Systems Society, Sept. 2006.

[8] A. J. Blumberg, L. S. Keeler, and abhi shelat. Automated traffic enforcement which respects 'driver privacy'. In S. Stramigioli, editor, *Proceedings of ITSC 2005*, pages 941–946. IEEE Intelligent Transportation Systems Society, Sept. 2005.

[9] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *Proceedings of PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, Jan. 2003.

[10] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[11] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Proceedings of Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer-Verlag, 2000.

[12] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Proceedings of Crypto 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer-Verlag, 2006.

[13] J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In *Proceedings of PKC 2009*, pages 196–214, 2009.

[14] CAMP Vehicle Safety Communications Consortium. Vehicle safety communications project task 3 final report, Mar. 2005. Online: http://www.intellidriveusa.org/documents/vehicle-safety.pdf.

[15] D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. In *Proceedings of Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145. Springer-Verlag, 2008.

[16] City-Data.com. San diego county, california detailed profile. http://www.city-data.com/county/San_Diego_County-CA.html.

[17] City News Service. Funding approved for survey of San Diego's road conditions. http://lajollalight.com/2011/01/11/funding-approved-for-survey-of-san-diegos-road-conditions/.

[18] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Proceedings of Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer-Verlag, 2002.

[19] W. de Jonge and B. Jacobs. Privacy-friendly electronic traffic pricing via commits. In P. Degano, J. Guttman, and F. Martinelli, editors, *Proceedings of FAST 2008*, volume 5491 of *LNCS*, pages 143–61. Springer-Verlag, 2009.

[20] R. Dingledine and N. Mathewson. Tor: the second-generation onion router. In *Proceedings of USENIX Security 2004*, pages 303–320, 2004.

[21] J. Freudiger, M. H. Manshaei, J.-P. Hubaux, and D. C. Parkes. On non-cooperative location privacy: a game-theoretic analysis. In S. Jha and A. D. Keromytis, editors, *Proceedings of CCS 2009*, pages 324–337. ACM Press, Nov. 2009.

[22] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Proceedings of Crypto 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 1997.

[23] GMP. The GNU MP Bignum library. http://gmplib.org.

[24] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[25] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of 17th Symposium on the Theory of Computing (STOC)*, pages 186–208, 1985.

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

[27] M. Green. *Cryptography for secure and private databases: enabling practical database access with-*

out compromising privacy. PhD thesis, Johns Hopkins University, 2009.

[28] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *Proceedings of Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 265–282. Springer-Verlag, 2007.

[29] J.-H. Hoepman and G. Huitema. Privacy enhanced fraud resistant road pricing. In J. Berleur, M. D. Hercheui, L. M. Hilty, and W. Caelli, editors, *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*, volume 328 of *IFIP AICT*, pages 202–13. Springer-Verlag, Sept. 2010.

[30] B. Hoh, M. Gruteser, R. Herring, and J. Ban. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of MobiSys 2008*, pages 15–28. ACM Press, June 2008.

[31] J.-P. Hubaux, S. Căpkun, and J. Luo. The security and privacy of smart vehicles. *IEEE Security & Privacy*, 2(3):49–55, 2004.

[32] S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *Proceedings of the 6th Theory of Cryptography Conference (TCC)*, pages 577–594, 2009.

[33] J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6): 391–399, Aug. 2009.

[34] H. Lipmaa. On Diophantine complexity and statistical zero-knowledge arguments. In *Proceedings of Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415. Springer-Verlag, 2003.

[35] S. Meiklejohn, C. C. Erway, A. Küpçü, T. Hinkle, and A. Lysyanskaya. ZKPDL: a language-based system for efficient zero-knowledge proofs and electronic cash. In *Proceedings of USENIX Security 2010*, pages 193–206, 2010.

[36] H. Peterson. Police chief denounces 'cowardly' iPhone users monitoring speed traps. *The Washington Examiner*, July 7, 2009.

[37] Photo Enforced. Crowdsourcing red light camera locations. Online: `http://www.photoenforced.com/`, 2004.

[38] R. A. Popa, H. Balakrishnan, and A. Blumberg. VPriv: protecting privacy in location-based vehicular services. In F. Monrose, editor, *Proceedings of USENIX Security 2009*, pages 335–50. USENIX, Aug. 2009.

[39] S. Rass, S. Fuchs, M. Schaffer, and K. Kyamakya. How to protect privacy in floating car data systems. In Y.-C. Hu and M. Mauve, editors, *Proceedings of VANET 2008*, pages 17–22. ACM Press, Sept. 2008.

[40] M. Scott. The MIRACL library. `http://www.shamus.ie`.

[41] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of Crypto 1984*, volume 7 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.

[42] C. Troncoso, G. Danezis, E. Kosta, and B. Preneel. PriPAYD: privacy friendly pay-as-you-drive insurance. In T. Yu, editor, *Proceedings of WPES 2007*, pages 99–107. ACM Press, Oct. 2007.

[43] United States Census Bureau. U.s. census bureau population estimates. `http://www.census.gov/popest/counties/tables/CO-EST2009-01-06.csv`.

[44] L. Vincentric. Law enforcement lifecycle cost analysis. http://vincentric.com/Portals/0/Market

A Blind and Anonymous IBE

In this section, we will see a formal outline of the blind IBE scheme we use (discussed briefly in Section 3), as well as the notion of security we expect it to satisfy and a proof that it does so. We start with the scheme, which as mentioned is essentially a "twin" version of the Boneh-Franklin IBE. Recall that, formally, a blind IBE consists of four algorithms: Setup, which is run by the authority to generate the master secret key and the system parameters; BlindExtract, which is run as an interaction between a user and the authority wherein the user obtains the secret key for a given identity (but without the authority learning which identity); IBEnc, in which a user can encrypt a message for the person specified by some identity; and finally IBDec, in which the user with the secret key for an identity can decrypt a ciphertext intended for him. We describe these algorithms for our scheme below:

- $\mathsf{Setup}(1^k)$: Compute a bilinear group G of prime order q with a generator g, as well an associated symmetric bilinear map $e : G \times G \to G_T$. Furthermore, set the message space to be $\mathcal{M} = \{0,1\}^m$ for some value of m. Pick random $x_1, x_2 \leftarrow \mathbb{F}_q^*$ and compute $X_1 = g_1^x$ and $X_2 = g_2^x$. Set $msk = (x_1, x_2)$ and $params = (q, G, G_T, g, e, \mathcal{M}, X_1, X_2, H, H')$, where H and H' are hash functions that map onto G and $\mathcal{M}$ respectively, and output *params*.

- $\mathsf{BlindExtract}(\mathsf{User}(params, id) \leftrightarrow \mathsf{Auth}(msk))$: For a given identity *id*, the user will start by picking randomness $r \leftarrow \mathbb{F}_q^*$ and sending $req = H(id) \cdot g^r$ to the authority. The authority then sends back $sk_1' = req^{x_1}$ and $sk_2' = req^{x_2}$. To complete the interaction, the user will use his randomness to compute $sk_1 = sk_1'/X_1^r$ and $sk_2 = sk_2'/X_2^r$. He can then check that these keys are properly formed by checking that $e(sk_1, g) = e(H(id), X_1)$ and $e(sk_2, g) = e(H(id), X_2)$.

- $\mathsf{IBEnc}(params, id, m)$: First pick randomness $r \leftarrow \mathbb{F}_q^*$. Next, compute intermediate values $Z_1 = e(H(id), X_1)^r$ and $Z_2 = e(H(id), X_2)^r$. Then, compute $h = H'(id, g^r, Z_1, Z_2)$ and output $c = (g^r, h \oplus m)$.
- $\mathsf{IBDec}(params, sk_{id}, c)$: Parse $c = (c_1, c_2)$ and $sk_{id} = (sk_1, sk_2)$. Compute intermediate values $Z_1 = e(sk_1, c_1)$, $Z_2 = e(sk_2, c_1)$, and $h = H'(id, c_1, Z_1, Z_2)$. Finally, output the message as $m = h \oplus c_2$.

Informally, we require that two security properties hold: first, in the process of running the BlindExtract protocol, even an adversarial user can learn nothing beyond his secret key; and, second, even an adversarial authority can learn nothing about a user's identity. In the original paper by Green and Hohenberger [28], these two properties are formalized as *leak-free extraction* and *selective-failure blindness* respectively. While the definition for selective-failure blindness is game-based, the definition for leak-free extraction is simulation-based, as they require (for their final application of simulatable oblivious transfer) that the identity be extractable from the BlindExtract protocol. As we do not need such a strong property, we propose an alternative and natural game-based definition of leak-free extraction, modeled after the traditional blind signature notion of one-more unforgeability first formalized by Pointcheval and Stern [**?**]. Our new notion, which we call *one-more indistinguishability*, is defined using the following game.

Definition A.1. For a blind IBE scheme $(\mathsf{Setup}, \mathsf{BlindExtract}, \mathsf{IBEnc}, \mathsf{IBDec})$, a given adversary $\mathcal{A}$, and bits $b_1, \ldots, b_{n+1} \leftarrow \{0,1\}$ unknown to $\mathcal{A}$, define the following game:

Step 1. $(msk, params) \leftarrow \mathsf{Auth}(1^k)$.

Step 2. $\mathcal{A}$ now gets access to *params* and to a blind extraction oracle. In other words, $\mathcal{A}$ will get to act as the user in an interaction with $\mathsf{Auth}(msk)$ n_0 times so that, for identities $id_1, \ldots, id_{n_0}$, $\mathcal{A}$ ends up with $sk_1, \ldots, sk_{n_0}$ (and the authority learns nothing about which keys were extracted).

Step 3. $\{(id_i, m_0^{(i)}, m_1^{(i)})\}_{i=1}^{n+1} \leftarrow \mathcal{A}(params, \{sk_i\}_{i=1}^{n_0})$.

Step 4. $\mathcal{A}$ now gets access to the set $\{c_i = \mathsf{IBEnc}(params, id_i, m_{b_i}^{(i)})\}_{i=1}^{n+1}$. $\mathcal{A}$ may now interact with the blind extraction oracle $n - n_0$ more times to end up with secret keys $sk_{n_0+1}, \ldots, sk_n$ for identities $id_{n_0+1}, \ldots, id_n$.

Step 5. Finally, $\mathcal{A}$ must output bits $b'_1, \ldots, b'_{n+1}$.

We say that the blind IBE scheme has *one-more indistinguishability* if for all such PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that for all $k > k_0$ the probability (over the choices of the b_i and the randomness used in Setup, BlindExtract, IBEnc, and $\mathcal{A}$) that $b'_i = b_i$ for all i, $1 \le i \le n+1$, is at most $1/2 + \nu(k)$.

In some blind extraction protocols (including ours) the user's messages hide his desired identity information-theoretically. Such protocols are not extractable, so they cannot satisfy the leak-free extraction notion of Green and Hohenberger. By contrast, our notion of one-more indistinguishability does not rule out such protocols.

(It would be interesting to construct blind extraction protocols for anonymous IBEs other than Boneh-Franklin, since these would avoid reliance on the random oracle model and could satisfy Green and Hohenberger's stronger notion of leak-free extraction. However, this appears to be a nontrivial undertaking. Indeed, the anonymous IBEs and hierarchical IBEs in the standard model of which we are aware obtain their anonymity precisely by withholding from users the values necessary to rerandomize their keys or to form a blinded Boneh-Boyen hash of their identity, which is necessary to avoid leaking information from the secret key or the extraction request, respectively.)

Using Green and Hohenberger's notion of selective-failure blindness and our notion of one-more indistinguishability, we can prove our blind Boneh-Franklin scheme secure. The proof, like the proof of Boneh-Franklin itself, is in the random oracle model. Because of the close relation between it and the Boldyreva blind signature [9], we cannot hope to prove our scheme secure under a nicer assumption than Boldyreva's chosen-target CDH assumption. The assumption we use, described below, is essentially the Bilinear Diffie-Hellman analogue of Boldyreva's chosen-target CDH.

Assumption A.2. For a prime-order bilinear group (q, G, G_T, g, e), a random value $x \leftarrow \mathbb{F}_q^*$, a "target" oracle that on any input outputs a random pair $(W, Y) \in G^2$, and a "helper" oracle that, on input $U \in G$, outputs U^x, it is hard for an adversary $\mathcal{A}$ that is given $X = g^x$ and access to these two oracles to output $\{e(W_i, Y_i)^x\}_{i=1}^{k+1}$, where k is the number of times it has queried the helper oracle.

Finally, we can put everything together and prove the following theorem:

Theorem A.3. *If Assumption A.2 holds, the IBE scheme outlined above is blind and anonymous in the random oracle model.*

As the anonymity of the scheme follows directly from the anonymity of the original Boneh-Franklin scheme, we can focus solely on the properties needed for blindness: one-more indistinguishability, as defined by the game-based Definition A.1 above, and selective-failure blindness, as defined by Green and Hohenberger [28, Definition 3].

Lemma A.4. *If Assumption A.2 holds, the scheme outlined above has one-more indistinguishability in the random oracle model.*

Proof. To show this, we need to take an adversary $\mathcal{A}$ that wins at the game in Definition A.1 with some non-negligible advantage ε and use it to construct an adversary $\mathcal{B}$ that breaks Assumption A.2 with some non-negligible probability ε'. Our adversary $\mathcal{B}$ will get as input the description (q, G, G_T, g, e) of some bilinear group and a value $X \in G$. To set up the parameters for the IBE, it will set $X_1 = X$ and then compute $X_2 = g^s / X^r$ for some random values $r, s \leftarrow \mathbb{F}_q^*$; note that this means the discrete logs of X_1 and X_2 (with respect to g) is are $x_1 = x$ and $x_2 = s - x_1 r$, neither of which $\mathcal{B}$ knows. $\mathcal{B}$ can now publish the parameters of the system as $(q, G, G_T, g, e, \mathcal{M}, X_1, X_2, H, H')$ (where $\mathcal{M}$ is just some arbitrary message space and H and H' are modeled as random oracles) and remember the values of r and s for later.

Now, $\mathcal{B}$ will need to answer three types of queries from $\mathcal{A}$: queries to H, queries to H', and blind extraction queries. Random oracle queries for H' will be very simple in this phase: on a query of the form (id, c_1, Z_1, Z_2), if $\mathcal{B}$ has not seen this query before then it will pick a random value $R \leftarrow \mathcal{M}$, set $H'(id, c_1, Z_1, Z_2) = R$, and both return this to $\mathcal{A}$ and remember it for later; if it has seen this query it will simply return the stored value. For H queries, $\mathcal{B}$ will get a query of the form id. If this is a new query then $\mathcal{B}$ can query its target oracle to get back a pair (W, Y); it then sets $H(id) = Y$, remembers both W and Y for later, and returns Y to $\mathcal{A}$. As with H' queries, if this is not a new query then $\mathcal{B}$ will simply return the value of Y it already has stored. Finally, for blind extraction queries, $\mathcal{B}$ will get a request req from $\mathcal{A}$. $\mathcal{B}$ can now query its helper oracle on req to get back $A = req^x$; it then sets $sk'_1 = A$ and $sk'_2 = req^s / A^r$, and returns (sk'_1, sk'_2) to $\mathcal{A}$.

Next, $\mathcal{A}$ will output the challenge set $\{(id_i, m_0^{(i)}, m_1^{(i)})\}_{i=1}^{n+1}$. $\mathcal{B}$ can now proceed as follows: For each tuple $(id_i, m_0^{(i)}, m_1^{(i)})$, $\mathcal{B}$ can pick a bit $b_i \leftarrow \{0,1\}$ and remember it for the next phase. If it has previously been queried (as the H oracle) on id_i, then it will set $c_{1i} = W_i$, where the response from the target oracle on query id_i was (W_i, Y_i) and it had previously set $H(id_i) = Y_i$. If not, it will query the target oracle to get the pair (W_i, Y_i); it will then set $c_{1i} = W_i$ and remember that $H(id_i) = Y_i$ for later. It can then pick a random string $R_i \leftarrow \mathcal{M}$ and set $c_{2i} = R_i$ (and again remember this for later). After repeating this process for each i, $\mathcal{B}$ can now return the set $\{c_i = (c_{1i}, c_{2i})\}_{i=1}^{n+1}$ to $\mathcal{A}$.

In the next phase, $\mathcal{B}$ can continue to answer hash queries to H and blind extraction queries in the same way as before. To answer H' queries now, on a given query (id, c_1, Z_1, Z_2) there are a number of possible choices. If $\mathcal{B}$ has seen this query before, then it will simply return the stored value. If $id \neq id_i$ for any i, $1 \leq i \leq n+1$ (in other words it is none of the identities chosen in the previous phase by $\mathcal{A}$), then $\mathcal{B}$ will again choose a random string R and set $H'(id, c_1, Z_1, Z_2) = R$. If id is in fact one of these identities, say $id = id_j$ for some j, then $\mathcal{B}$ can now look at c_1. If $c_1 \neq W_j$, then $\mathcal{B}$ can again run the process of setting the hash value to be random. If $c_1 = W_j$, then $\mathcal{B}$ will check that $Z_1^r Z_2 = e(Y_j, W_j)^s$. If this outputs 0, then $\mathcal{B}$ can simply set the hash value to be random as before. If, on the other hand, this outputs 1, then $\mathcal{B}$ will use the chosen bit b_j from before and set $H'(id, c_1, Z_1, Z_2) = m_{b_j}^{(j)} \oplus c_{2j}$. $\mathcal{B}$ will then return this hash value to $\mathcal{A}$ and store the value Z_1 for itself.

At the end, $\mathcal{A}$ will output its guess bits $\{b'_i\}_{i=1}^{n+1}$. If $\mathcal{A}$ successfully guesses the correct bit for each i with non-negligible advantage ε, then we can argue that with non-negligible probability ε' $\mathcal{B}$ will have accumulated in the course of the above interaction $n+1$ values of the form $e(W_i, Y_i)^x$, where x is the discrete log of its original input X. To see this, note that the only case in which $\mathcal{B}$ returned a non-random value for an H' query was when the query (id, c_1, Z_1, Z_2) was "well-formed" in the sense that id and c_1 matched one of the above ciphertexts, and Z_1 and Z_2 were formed honestly with respect to the decryption process. To see this last part, note that the test $Z_1^r Z_2 = e(Y_j, W_j)^s$ is just the Twin BDH test of Cash, Kiltz, and Shoup [15]. If Z_1 and Z_2 were in fact properly formed, then we have

$$\begin{aligned} Z_1^r Z_2 &= e(Y^{x_1}, W)^r \cdot e(Y^{x_2}, W) \\ &= e(Y, W)^{rx_1 + x_2} \\ &= e(Y, W)^{rx + (s - rx)} \\ &= e(Y, W)^s, \end{aligned}$$

so that this equality will hold and the test will pass. Conversely, if Z_1 and Z_2 are not properly formed, the check performed by $\mathcal{B}$ will fail with overwhelming probability (over the choice of r and s). To see this, we rewrite $Z_1^r Z_2 = e(Y, W)^s$ as

$$\left(Z_1 / e(Y, W)^{x_1}\right)^r = e(Y, W)^{x_2} / Z_2 \tag{2}$$

(using the fact that $x_2 = s - x_1 r$). IF Z_1 and Z_2 are not both properly formed then at least one of $Z_1 / e(Y, W)^{x_1}$ and $Z_2 / e(Y, W)^{x_2}$ is not equal to 1; but then the probability that Equation (2) holds is zero if $Z_1 / e(Y, W)^{x_1} = 1$ and $1/q$ otherwise, since r is independent of the adversary's view. Since $\mathcal{A}$ can make only polynomially many queries to H', then, by the union bound, the probability that $\mathcal{B}$ ever accepts an improperly formed tuple (Z_1, Z_2) is still negligible. Provided this never happens, unless $\mathcal{A}$ queries H' with the correct (Z_1, Z_2) pair, the message remains information-theoretically hidden, as the value c_2 is completely random and independent from the message and the value of $H'(id, c_1, Z_1, Z_2)$ will be as well. So, the only

way for $\mathcal{A}$ to learn any information at all about the message (and thus gain non-negligible advantage in guessing the bit) is to send an H' query to $\mathcal{B}$ in which Z_1 and Z_2 are well-formed, so Z_1 looks like $e(Y^{x_1}, W) = e(Y,W)^{x_1}$; because we set $X_1 = X$ this is furthermore equal to $e(Y,W)^x$, which is exactly the type of value $\mathcal{B}$ is looking for.

Therefore, for every bit that $\mathcal{A}$ guesses correctly with non-negligible advantage over $1/2$, $\mathcal{B}$ must have also obtained a tuple of the form $e(Y,W)^x$ with similar non-negligible probability. If $\mathcal{A}$ does not query H' at the appropriate value for even one of the $n+1$ bits it must guess, then the probability that it succeeds in guessing all of them is $1/2$, since that unqueried bit remains hidden. Thus if A has non-negligible advantage in guessing all $n+1$ bits, it must reveal to $\mathcal{B}$ values $Z = e(Y,W)^x$ for each of $n+1$ pairs (W,Y) obtained from $\mathcal{B}$'s target oracle.

Finally, we need to argue that the interaction with $\mathcal{B}$ is indistinguishable from the interaction $\mathcal{A}$ would have in the honest game. As H and H' are random oracles and $\mathcal{B}$ remembers answers to old queries to maintain consistency, the only case we must consider is when $\mathcal{B}$ doesn't pick the output uniformly at random and instead sets $H'(id_j, c_{1j}, Z_1, Z_2) = R_j \oplus m_{b_j}^{(j)}$ for some j. As R_j is uniformly random though, this will also be random and so the distribution of the answers that $\mathcal{B}$ gives for both the H and H' queries is identical to the honest distribution. Similarly, for blind extraction $\mathcal{B}$ in fact computes the keys honestly, as it returns $sk'_1 = req^{x_1}$ and $sk'_2 = req^s/(req^{x_1})^r = req^{s-x_1 r} = req^{x_2}$, which is exactly what an honest authority would return.

Next, we consider the distribution of the ciphertexts $\{c_i\}$. In the honest case we have $c_{1i} = g^{r_i}$ for some uniformly random value r_i; as the W_i output by the target oracle is also uniformly random and $\mathcal{B}$ uses $c_{1i} = W_i$, we have that the distribution over these values is again identical. Similarly, $\mathcal{B}$ uses $c_{2i} = R_i$ instead of $c_{2i} = h \oplus m$; again though, because h is assumed to be random, both of these values will be distributed uniformly at random over $\mathcal{M}$. Finally, we note that when $\mathcal{B}$ does use $h_j = H'(id_j, c_{1j}, Z_1, Z_2) = c_{2j} \oplus m_{b_j}^{(j)}$ the ciphertext will correctly decrypt as $h_j \oplus c_{2j} = m_{b_j}^{(j)}$. In all other cases this will return some random value unrelated to the message (because the hash value will just be random); this would happen in the honest case as well, however, as $\mathcal{B}$ answers H' queries randomly for the "special" choices of id_j and c_{1j} only if Z_1 and Z_2 are malformed (and in the non-special cases its answers are still consistent, as $\mathcal{A}$ would need to make H' queries both to encrypt and decrypt). Note that, except with negligible probability, $\mathcal{A}$ cannot query H' at a properly-formed point (id_j, c_{1j}, Z_1, Z_2) before it has seen the challenge ciphertexts, as it will not have seen the values $\{c_{1j}\}$, which are uniformly distributed in an exponentially large space.

As we've now argued that $\mathcal{A}$ cannot distinguish between playing the honest game and playing the game with $\mathcal{B}$ (in fact, we've shown that there is no difference, except with negligible probability), we know that $\mathcal{A}$ will behave identically in both cases. Furthermore, as $\mathcal{B}$ succeeds whenever $\mathcal{A}$ succeeds (except with negligible probability) and $\mathcal{A}$ was assumed to succeed with non-negligible advantage, we have successfully constructed our adversary $\mathcal{B}$ to break Assumption A.2 with non-negligible probability and so we are done. □

Now that we have shown one-more indistinguishability, we only need to show that selective-failure blindness holds as well and we will be done. Fortunately, proving selective-failure blindness is much easier, as essentially all the values in our scheme (the keys, ciphertexts, etc.) are information-theoretically indistinguishable from the identity being used, meaning the identity is always unconditionally hidden.

Lemma A.5. *Any adversary, even one that is computationally unbounded, will have no advantage in the selective-failure blindness game for our IBE scheme.*

Proof. We can start by reminding ourselves of the game that $\mathcal{A}$ will play: it can first pick the IBE parameters *params* and two identities id_0 and id_1; it then engages in the BlindExtract protocol with two honest users, one using id_b and the other using id_{1-b} for some bit $b \leftarrow \{0,1\}$ unknown to $\mathcal{A}$. If neither user outputs an error message (indicating that the resulting sk_b and sk_{1-b} values were well-formed) then $\mathcal{A}$ gets to see both keys; otherwise, $\mathcal{A}$ gets to learn which one failed but no information about the one that succeeded. This means there are three potential places for $\mathcal{A}$ to learn information about the bit b: the protocol interaction itself, whether or not the users output error messages, and the final secret keys if it is given them. We can consider each of these options argue in turn that each of them is not possible.

Protocol interaction. Our BlindExtract protocol consists of two messages, which means that the only opportunity for $\mathcal{A}$ to learn anything about the underlying identity is through the query value *req*. Observe, however, that for the user with id_b we have $req = H(id_b) \cdot g^r$ for some uniformly random $r \leftarrow \mathbb{F}_q^*$. This *req* value will therefore be information-theoretically independent from the identity id_b, as it will just be a uniformly random element of G. As the same holds true for the user with id_{1-b}, $\mathcal{A}$ cannot hope to gain information about b from the interaction itself.

Whether users accept. Looking back at the scheme, we can see that the user with id_b will accept the values sk'_{b1} and sk'_{b2} given to him by $\mathcal{A}$ if and only if $e(sk'_{b1}/X_1^r, g) = e(H(id_b), X_1)$ and $e(sk'_{b2}/X_2^r, g) = e(H(id_b, X_2)$. Now, we can argue that $\mathcal{A}$ can in fact determine whether or not these equalities hold on its own, and will thus learn no information from this step. To see this, we can focus on the case for sk_{b1}; the argument will be identical for sk_{b2}. Because $\mathcal{A}$ computed sk'_{b1} it clearly knows this value, as well as the discrete log x_1 of X_1. It can then compute $A = req^{x_1} = H(id_b)^{x_1} \cdot g^{x_1 r}$ and check that

$$e(sk'_{b1}, g) = e(A, g). \tag{3}$$

Using the bilinearity property of the pairing, we can see that

$$\begin{aligned} e(A,g) &= e(H(id_b)^{x_1} \cdot g^{x_1 r}, g) \\ &= e(H(id_b)^{x_1}, g) \cdot e(g^{x_1 r}, g) \\ &= e(H(id_b), g)^{x_1} \cdot e(X_1^r, g) \\ &= e(H(id_b), g^{x_1}) \cdot e(X_1^r, g) \\ &= e(H(id_b), X_1) \cdot e(X_1^r, g). \end{aligned}$$

Dividing both sides of Equation 3 by $e(X_1^r, g)$ therefore gives us $e(sk'_{b1}, g)/e(X_1^r, g) = e(H(id_b), X_1)$, or $e(sk'_{b1}/X_1^r, g) = e(H(id_b), X_1)$, which we observe is the exact check run by the user. The check done by $\mathcal{A}$ will therefore pass if and only if the user's check will pass, which means the adversary can determine no information about b from this stage, as any information learned from the users' acceptance could have already been determined by $\mathcal{A}$ itself.

The resulting secret keys. Finally, if both users accept their secret keys then $\mathcal{A}$ now has access to $sk_0 = (H(id_0)^{x_1}, H(id_0)^{x_2})$ and $sk_1 = (H(id_1)^{x_1}, H(id_1)^{x_2})$. Note that these values are not randomized, and depend only on id_0, id_1, x_1, and x_2. As these are all values provided by $\mathcal{A}$, however, it can learns nothing from seeing sk_0 and sk_1, as it could have computed them itself.

We are now done, as we have argued that $\mathcal{A}$ can learn no information whatsoever about the identities in any part of the blind extraction process, even if $\mathcal{A}$ is in fact computationally unbounded. □