Privacy-Preserving Observation in Public Spaces

Florian Kerschbaum[1(✉)] and Hoon Wei Lim[2]

[1] SAP, Karlsruhe, Germany
florian.kerschbaum@sap.com
[2] Singtel R&D Laboratory, Singapore, Singapore
limhoonwei@singtel.com

Abstract. One method of privacy-preserving accounting or billing in cyber-physical systems, such as electronic toll collection or public transportation ticketing, is to have the user present an encrypted record of transactions and perform the accounting or billing computation securely on them. Honesty of the user is ensured by spot checking the record for some selected surveyed transactions. But how much privacy does that give the user, i.e. how many transactions need to be surveyed? It turns out that due to collusion in mass surveillance *all* transactions need to be observed, i.e. this method of spot checking provides no privacy at all. In this paper we present a cryptographic solution to the spot checking problem in cyber-physical systems. Users carry an authentication device that authenticates only based on fair random coins. The probability can be set high enough to allow for spot checking, but in all other cases privacy is perfectly preserved. We analyze our protocol for computational efficiency and show that it can be efficiently implemented even on platforms with limited computing resources, such as smart cards and smart phones.

1 Introduction

Cyber-physical systems are starting to permeate our daily lives. They record time and location information – together with sensory data – of ourselves and this data is, in turn, used to analyze our behavior in the physical world. A common application of such cyber-physical systems is billing, e.g. for toll collection [2, 22, 26], public transportation [11, 17, 28], or electric vehicle charging [21]. The cyber-physical sensor records our identity, time, location and consumption. This data is then used to bill us based on the recorded transactions.

An obvious problem with this approach is privacy. All our transactions in the physical world are recorded and can be also analyzed for purposes other than billing. Every search engine or web-based e-mail user already gets displayed a huge amount of personalized ads.

Instead of centrally collecting all transactions they can be stored on user-owned devices. The user then presents its collected record of transactions and pays its bill. While this would remove the central data storage, the service provider observes all transactions during payment and could theoretically retain a copy. An approach for protecting privacy is to have the user present an

G. Pernul et al. (Eds.): ESORICS 2015, Part II, LNCS 9327, pp. 81–100, 2015.
DOI: 10.1007/978-3-319-24177-7_5

encrypted record of transactions. A computation on the encrypted transactions then results in the billing amount. No information – in the clear – is revealed during this process. This approach has been taken in [2,11,17,21,22,26,28]. Similar approaches can be realized using secure multi-party computation [4,5,13–16,18,19,30].

The problem is, of course, that the billed person could cheat and present an incomplete, tampered or even empty record of transactions. A solution is to record some transactions of the user in the physical world and spot check whether he honestly included them in the presented transactions during bill payment. Popa et al. [26] and Balasch et al. [2] recently presented systems for road toll pricing that follow this model. Yet, the problem is still not solved, since the users may collude in order to determine where and when they have been observed. Meiklejohn et al. [22] therefore proposed to keep the spots that are checked secret.

A problem that arises in the approach of [22] is that collusion among *dishonest* users is still possible, even if the spot checks are kept secret. A dishonest user may risk paying the penalty, if he gains from the information obtained. If this user is caught, he must be presented evidence that he cheated which reveals the observed spot. Dishonest users may therefore collude by submitting incomplete records of transactions and share the obtained information and the penalties for getting caught. We show in Sect. 3 that mass surveillance cannot prevent this collusion under reasonable penalties and *all* transactions need to be observed in order to prevent collusion. Clearly, this method then provides no privacy at all. Moreover, we argue that the proposal of [22] does not solve the problem of enforcing honesty vs. privacy. The question one has to ask is how many transactions need to be surveyed. Too few transactions may enable cheating and too many violate privacy. We investigate whether there is a privacy-compliant trade-off in public spot checking.

We propose a cryptographic solution to this problem. Particularly, we present a protocol for a privacy-preserving spot checking device. The basic idea is to run a randomized oblivious transfer of the trace of an authentication protocol and a random message. This device may authenticate the carrier (due to the authentication protocol) with a random probability that cannot be tweaked by the carrier (if the oblivious transfer sends the authentication trace). The observer learns nothing with a probability that cannot be tweaked by the reader (if the oblivious transfer sends the random message). The probability of authentication can be set high enough to allow for spot checking, but low enough to provide reasonable privacy. Since it is a personal device authenticating only its carrier, no information can be shared, and thus completely preventing collusion attacks.

We emphasize that the construction of our protocol is very efficient. We neither need any secure hardware nor verifiable encryption using zero-knowledge proofs. We achieve this by using a definition of fairness secure against one-sided malicious adversaries only restricting the attacks to incentive-compatible ones.

All our security objectives are provably achieved using efficient, established building blocks and assumptions. A secure device could be built from the specification by anyone – even suspicious citizens. Furthermore, we optimize the protocol for performance and present an implementation analysis of the optimized version for weak computation devices. We estimate that the protocol can be run in roughly half a second on very weak smart cards. As a conclusion it is likely that the protocol can be run even on flowing traffic without any disruptions.

In this paper we contribute

- an economic analysis of collusion in spot checked cyber-physical systems;
- a protocol for privacy-preserving spot checking;
- an implementation analysis of this protocol for weak computational devices.

The remainder of this paper is structured as follows. In the next section, we give a brief overview of related work on privacy-preserving electronic billing or accounting systems. In Sect. 3 we present our economic analysis of collusion in these systems. Then, we describe our privacy-preserving spot checking protocol in Sect. 4. We also give an implementation analysis and discuss the trade-off between privacy and enforcement of honesty. In Sect. 5 we give an example application and we present our conclusions in Sect. 6.

2 Related Work

2.1 Privacy-Preserving Billing

Toll Collection. Cryptographic privacy-preserving toll collection systems [2, 22, 26] have been proposed to resolve the tension between the desire for sophisticated road pricing schemes and drivers' interest in maintaining the privacy of their driving patterns. At the core of these systems is a monthly payment and an audit protocols performed between the driver, via an on-board unit (OBU), the toll service provider (operating the OBU) and the local government. Each driver commits to the road segments she traversed over the month and the cost associated with each segment. To ensure honest reporting, the systems rely on the audit protocol, which in turn, makes use of unpredictable spot checks by hidden roadside cameras. At month's end, the driver is challenged to show that her committed road segments include the segments in which she was observed, and that the corresponding prices are correct. (More description of such an audit protocol for privacy-preserving toll collection is given in Sect. 5.) As long as the spot checks are done unpredictably, any driver attempting to cheat will be caught with high probability. Meiklejohn et al. [22] proposed a system called Milo, which employs an oblivious transfer technique based on blind identity-based encryption [9] in its audit protocol, such that spot checks on a driver's committed road segments can be performed without revealing the checked locations. Nevertheless, privacy is preserved with respect to only the toll service provider. The cameras can actually observe *all* vehicles at all times, and thus, there is completely no privacy against the local government (audit authority), which can trace any vehicle and learn its driving patterns through the cameras.

e-Ticketing in Public Transportation. The use of contactless smart cards as electronic tickets (e-tickets) is popular in public transportation systems in many countries world-wide [11,17,28]. However, majority of existing e-ticketing systems are not designed to protect user privacy, i.e. travel records leak commuters' location information. In fact, transportation companies collect commuters' travel history in order to analyze traffic patterns and detect fraudulent transactions. However, this clearly is a privacy breach. Kerschbaum et al. [17] recently proposed a cryptographic solution for bill processing of travel records while allowing privacy-preserving data mining and analytics. However, privacy is achieved at the expense of high processing overhead of encrypted travel records and the need for an independent key management authority. Also, no notion of spot checking is used in their system.

Electric Vehicle Charging. Electric vehicles have been used as a more environmental-friendly alternative to traditional gasoline-based vehicles. However, electric vehicles require frequent recharging at dedicated locations. If not done properly, information of a driver's whereabouts can be leaked through the underlying payment system. Liu et al. [21] designed an anonymous payment protocol for enhancing the location privacy of electric vehicles. While a driver's location privacy is preserved against the power grid company, a malicious local government is still able to reveal any past transactions.

Variants of Oblivious Transfer. Oblivious transfer is a protocol between a sender and a receiver. In its most simple form, the sender has two messages and the receiver learns one without learning anything about the other one. Oblivious transfer has been introduced by Rabin [27] where the message received was a joint random choice between sender and receiver. Even et al. generalized this to a setting where the receiver could choose which message he receives [7].

Oblivious transfer is a very powerful primitive. Kilian showed that all cryptographic primitives can be based on oblivious transfer [20]. Also many variants of oblivious transfer exist. In priced oblivious transfer [1] a price is deducted from an encrypted account for each message received. In k-out-of-n oblivious transfer [3] k messages can be chosen amongst n. We employ oblivious transfer in a new setting with authentication. Private information retrieval (PIR) is incompatible with our setting, since it may reveal information about the non-transferred message.

2.2 Threat Model

In privacy-preserving billing systems, the goal is to protect the location information of users while ensuring that the users behave in an honest manner.

We assume that there exist various *semi-honest* parties who may be interested in garnering information about users' travel patterns and location information; for example, service providers may do this for business reasons, local government for social or political reasons, and even users themselves may attempt to learn other users' whereabouts for malicious motives.

Moreover, there are *dishonest* users who may find every opportunity to cheat, including colluding with other users or deviating arbitrarily from the associated payment protocol. That is, they would attempt to avoid paying or paying less than they should for the services they have received.

3 Collusion Attack

Meiklejohn et al. [22] claim for their system Milo that it prevents collusion attacks, since it does not reveal the spot checked locations to an *honest* user. In this section we investigate the economic feasibility of a collusion attack for *dishonest* users. We consider a simplified model where the penalties and costs are linear in the number of transactions, but conjecture that similar solutions can be found in the non-linear case. Our conclusion is that in mass surveillance possible collusion leads to the need for observing all transactions.

3.1 Model

We divide *time* and *space* into discrete spots. Spots have a certain duration – usually as long as an observation period. Not all spots of a location must be observed, imagine, for example, mobile cameras.

A spot can be either observed, i.e. all transactions of users are recorded, or unobserved, i.e. no transaction is recorded. We assume that a fraction $\frac{1}{\alpha}$ of spots are to be observed. For the user the state of the spot may also be unknown, i.e. he does not know whether it is observed or unobserved.

During the duration of each spot *on average* m transactions of different users are observed. Imagine a camera that records the flowing car traffic and recognizes the license plates. In an hour on a busy street one can observe probably thousands of cars.

When the user reports his transactions, he has to pay cost d for each reported transaction. He may also choose to cheat and not report some transactions. For every transaction where he gets caught doing so he has to pay penalty p $(p > d)$.

3.2 Collusion Strategy

We consider the following strategy. For every spot where the user knows that he is unobserved, he does not report his transaction. For every spot where the user knows that he is observed, he, of course, reports, since $p > d$ and he will get caught. The question is whether he reports spots where the state is unknown to him.

For this we need to consider how a spot becomes known to be observed or unobserved. If a user does not report a spot and is charged with a penalty for that spot, the provider must present some evidence. From this evidence we can conclude that the spot is observed. If a user does not report a spot and is not charged with a penalty for that spot, he can likely conclude that the spot is unobserved.

We assume perfect information sharing about states of spots between all colluders, i.e. if one party is charged with a penalty all others know the spot is observed. We furthermore reward users for testing the states of the spots by not submitting transactions. If a user does not report a spot and is not charged with a penalty, he marks it as known unobserved and attaches his name to it. From now on, all other users also do not report this spot and pay a reward e ($e < d$) to this user. Clearly, these users will save some money compared to being honest. For each known observed spot they pay d, for each known unobserved spot they pay e. Since $e < d$, this saves them some money.

3.3 Analysis

The utility of reporting a transaction for a spot whose state is unknown is

$$U = \frac{1}{\alpha}p - (1 - \frac{1}{\alpha})me. \tag{1}$$

If $U > 0$, then the user reports honestly (and will not learn any information). We can compute the necessary penalty for discouraging dishonesty as

$$d > e \wedge p > (\alpha - 1)md \Rightarrow U > 0.$$

If we consider mass surveillance (say m on the order of thousands) and reasonable privacy (say α on the order of hundreds), the penalty p needs to be significantly (on the order of several hundreds of thousands) higher than the cost d for a transaction. Loosely speaking, this is roughly equivalent to a life sentence for getting caught riding on the bus without a ticket. Otherwise, it is rational to collude and cheat in our model.

Another solution to this problem is to observe every spot and transaction ($\alpha = 1$). This complete surveillance incentivizes honesty, but completely removes any privacy. The transactions of the user are known anyway, such that he does not need to report them in a privacy-preserving way, i.e. even the privacy-preserving measures are led ad absurdum. Our solution preserves privacy, but does single user privacy-preserving spot checking, i.e. $m = 1$. Surveillance monitors should still be at every spot, i.e. every spot is potentially observed, but the spot checking is random and cannot be forced.

Of course, the toll service provider could randomize spot checking by himself, but the user would have no guarantee and it is more economical for the provider to observe all spots. We eliminate this option of mass surveillance by proven privacy guarantees for the users.

$$\frac{P : s, SK, PK}{V : t} \xrightarrow{\Pi} \frac{P :}{V : z = s - t} \quad \text{if } z = 0 : PK$$

Fig. 1. Black-box protocol Π for privacy-preserving spot checking

4 Privacy-Preserving Spot Checking

We have a user who needs to be randomly spot checked. In the spirit of zero-knowledge proofs[1] we call him prover P. We have a provider who needs to perform the spot checking. We call him the verifier V.

Privacy-preserving spot checking is a protocol Π between P and V. P's inputs to Π are a secret key SK, a public identifier PK and a collection of random coins s. V's inputs to Π are random coins t. V's outputs is whether $z = s - t = 0$. If $z = 0$, V obtains the public identifier PK of the prover; otherwise, he obtains nothing more. P obtains no output – not even whether $z = 0$. Figure 1 displays the black-box input and output behavior of protocol Π.

4.1 Setup and Registration

There is a separate registration and verification phase. Each prover chooses a secret key SK or has it chosen for him by some authority. He then computes the public identifier PK.

We operate in some finite group $\mathbb{G}$ of prime order. Let v be the secret key. Then the public identifier is g^v. Clearly, we can assume that it is difficult to compute the secret key from the public identifier due to the hardness of the discrete logarithm problem. Since we only rely on the confidentiality of the secret key for authentication, we do not need any secure hardware in constructing our device.

The authority registers the personal identity of the prover along with public identifier g^v. Then, whenever presented with g^v the authority can personally identify the prover. Moreover, the authority can detect forged spot checking devices, if there is no record for a presented public identifier.

$$\begin{array}{ll}
\text{Input } P: & \text{(uniformly random) } r, \text{(secret key) } v \\
\text{Input } V: & \text{(uniformly random) } a \\
\text{Output } P: & - \\
\text{Output } V: & \text{(public identifier) } g^v, \mathit{accept/reject}
\end{array}$$

$$\begin{array}{l}
P \rightarrow V : g^v, g^r \\
V \rightarrow P : a \\
P \rightarrow V : r + av
\end{array}$$

Fig. 2. Protocol 1: standard schnorr identification

[1] Our protocol is not a full-fledged zero-knowledge protocol, but more efficient.

4.2 Security Properties

We demand a number of security properties from our privacy-preserving spot checking protocol.

Authenticity: In case $z = 0$, i.e. the verifier obtains public identifier PK, the prover cannot have produced PK without knowing the corresponding secret key SK. Hence, the prover is authenticated. We formalize this security notion equivalent to the soundness notion of zero-knowledge protocols. Let $\mathcal{E}^{P(PK)}$ be an extractor that given rewinding access to P extracts the secret key SK from P given that $z = 0$ and the public identifier PK has been revealed. We say a protocol is *authentic*, if

$$P \overset{\Pi}{\longleftrightarrow} V : z = 0, PK \Rightarrow 1 - \Pr[\mathcal{E}^{P(PK)} = SK] < 1/poly(\kappa).$$

Privacy: In case $z \neq 0$, i.e. the verifier does not obtain PK, the verifier cannot extract any information about the identity of the prover from the protocol. We formalize this security notion equivalent to zero-knowledge. Let $\mathcal{S}(s,t)^P$ be a simulator of the protocol view of the verifier – the messages of P – in case $s-t \neq 0$. We denote computational indistinguishability as $\overset{cind}{\sim}$. We say a protocol is *private*, if

$$\mathcal{S}^P(s,t) \overset{cind}{\sim} \Pi^P_{s-t\neq 0}.$$

Fairness: Let $V(\Pi)$ denote the verifier's output from the protocol, i.e. whether the prover cheated. The verifier V should only be able to force the case $z = 0$ with probability less than $\frac{1}{\alpha} + \frac{1}{poly(\kappa)}$. The prover P should only be able to force the case $z \neq 0$ with probability less than $1 - \frac{1}{\alpha} + \frac{1}{poly(\kappa)}$ without being detected. Let $\mathcal{A}^V$ and $\mathcal{A}^P$ be the adversary taking the role of the verifier and prover, respectively. We say a protocol is *fair*, if

$$Pr[P \overset{\Pi}{\longleftrightarrow} \mathcal{A}^V : z = 0] < \frac{1}{\alpha} + 1/poly(\kappa)$$

$$Pr[\mathcal{A}^P \overset{\Pi}{\longleftrightarrow} V : z \neq 0, V(\Pi) = 1] < 1 - \frac{1}{\alpha} + 1/poly(\kappa).$$

Reverse Unobservability: The prover P should not learn whether $z = 0$, i.e. whether he was observed or not. Let $\mathcal{S}^V$ be a simulator of the protocol view of the prover – the messages of V. We say a protocol is *reverse unobservable*, if

$$\mathcal{S}^V \overset{cind}{\sim} \Pi^V.$$

4.3 Protocol

We present the protocol in incremental steps in order to show the necessity and effect of individual messages. Let g denote a group generator. We begin with a regular Schnorr identification protocol [29] in Fig. 2.

The verifier accepts if $g^{r+av} = (g^v)^a g^r$, else he rejects. Clearly, this already achieves authenticity, since it is a standard Schnorr identification protocol. We can construct an extractor in the usual way.

$$\begin{array}{ll}
\text{Input } P: & \text{(uniformly random) } r, \text{(uniformly random) } s: 0 \leq s < \alpha, \text{(secret key) } v \\
\text{Input } V: & \text{(uniformly random) } t: 0 \leq t < \alpha \\
\multicolumn{2}{l}{\text{Output } P: z = s - t} \\
\multicolumn{2}{l}{\text{Output } V: z = s - t, \text{if } z = 0: \text{(public identifier) } g^v, accept/reject}
\end{array}$$

$$\begin{array}{l}
P \to V: Commit(s), Commit(g^v), Commit(g^r), r + H(g^r)v \\
V \to P: t \\
P \to V: Open(s) \\
P \to V: \text{if } z = s - t = 0: Open(g^v), Open(g^r)
\end{array}$$

Fig. 3. Protocol 2: schnorr identification with fair coin flip

Theorem 1. *Protocol 1 is* authentic.

Proof. The extractor proceeds as follows. It waits for the prover to send g^r. Then, it first sends a_1 and waits for $r + a_1 v$. Second, it rewinds P to the second protocol step and sends a_2. From $r + a_1 v$ and $r + a_2 v$, it computes v. Since the probability of randomly choosing a valid public identifier is negligible, the verifier can also verify the authenticity.

The protocol can be made completely non-interactive using the weak Fiat-Shamir heuristic [8]. The challenge a is then computed using a collision-resistant hash function $H(g^r)$.

Protocol 1 so far does not include a random choice, i.e. the identity of P is always revealed. The verifier receives g^v in the protocol. We now modify Protocol 1 to obtain Protocol 2. Here we incorporate a fair flip of a coin with probability $Pr[coin = 0] = 1/\alpha$ (and the Fiat-Shamir heuristic). Note that $r + av$ does not reveal the public identifier g^v, if also g^r is unknown. Therefore the prover commits to g^v and g^r using a cryptographic commitment scheme and only opens if the coin turns up $z = 0$.

A cryptographic commitment scheme consists of two operations:

- $\gamma = Commit(x, c)$: A commitment γ to x using the random coins c.
- $Open(\gamma) = x, c$: An opening of the commitment revealing the value x and the random coins c.

A cryptographic commitment scheme enjoys two properties:

- *Binding*: The probability that one can successfully open to a different x' is negligible.
- *Blinding*: The probability to compute x from γ is negligible.

A possible commitment scheme is that by Pedersen [25], but in practical implementations one can use, e.g. HMAC (secure in the random oracle model). For clarity of the description we leave out the random coins in the description of the commitment of the protocol, i.e. we write $Commit(x)$ and $Open(x)$. The protocol proceeds as in Fig. 3.

Theorem 2. *Protocol 2 is* private*.*

Proof. In case $z \neq 0$, the verifier receives the following messages $Commit(s)$, $Commit(g^v)$, $Commit(g^r)$, $r + H(g^r)v$ and $Open(s)$. The simulator proceeds as follows. It chooses random s and the corresponding coins and computes $Commit(s)$. It can already simulate two messages. It simulates $Commit(g^v)$ and $Commit(g^r)$ using two random commitments, since the commitments are never opened and blinding. It simulates $r + H(g^r)v$ using a random value, since g^r is unknown and hence $H(g^r)$ is pseudo-random.

Theorem 3. *Protocol 2 is* fair*.*

Proof. The protocol embeds a regular fair coin flip. The prover first chooses and commits to its value s and at the end of the protocol he opens the commitment. Due to the binding property of commitments, he must be honest or get caught, because either the commitment is not correctly opened or he chose an invalid value. He can only achieve $z \neq 0$ when $s \neq t$ or when he opens the commitment to a different value (in $[0, \alpha - 1]$). Hence the prover cannot force $z \neq 0$ with probability higher than $1 - \frac{1}{\alpha} + negl(\kappa)$.

The verifier chooses and sends his value t after the choice of s. He must do so without knowledge of s, since the commitment is blinding. He can only achieve $z = 0$ when $t = s$ and he has no influence on s. Hence, he cannot force $z = 0$ with probability higher than $\frac{1}{\alpha}$.

Note that the previous statement also holds when t is encrypted, since an invalid encryption of t can only achieve $z \neq 0$.

The problem with this protocol is that the prover still learns the outcome of the fair coin flip, i.e. he knows whether he is being observed. In our scenario this leads to the problem that he can now easily choose which transactions to report, since he knows all observed spots. We therefore introduce a technique we call *blind decommitment* where the prover opens (decommits) his commitment without knowing whether he does that. The decommitment is probabilistic and opening occurs only with probability $\frac{1}{\alpha}$, but the prover cannot influence this probability nor can he determine the type of event – decommitment or not.

We use a semantically secure and additively homomorphic encryption scheme, e.g. Paillier's [24], for this purpose. Let $E_V()$ denote the encryption in this homomorphic scheme under the verifier's key and $D_V()$ the corresponding decryption. Then the following homomorphism properties hold:

$$D_V(E_V(x)E_V(y)) = x + y$$

$$D_V(E_V(x)^y) = xy.$$

The verifier sends its random value t encrypted and the prover computes the decommitment homomorphically. We denote "*Open*(x)" as the encoding in $\mathbb{G}$ of the opening of x for homomorphic encryption. The prover needs to ensure that in case $z \neq 0$ the decommitment is safely blinded, i.e. using sufficient randomness although α is small. In case $z = 0$, the verifier can check the validity of the

$$\begin{array}{ll}
\text{Input } P: & \text{(uniformly random) } r, \text{(uniformly random) } u_1, u_2 \\
 & \text{(uniformly random) } s : 0 \leq s < \alpha, \text{(secret key) } v \\
\text{Input } V: & \text{(uniformly random) } t : 0 \leq t < \alpha \\
\text{Output } P: - & \\
\multicolumn{2}{l}{\text{Output } V : z = s - t, z = 0 : \text{(public identifier) } g^v, \mathit{accept}/\mathit{reject}}
\end{array}$$

$$\begin{array}{l}
P \to V : \mathit{Commit}(s), \mathit{Commit}(g^v), \mathit{Commit}(g^r), r + H(g^r)v \\
V \to P : E_V(-t) \\
P \to V : \mathit{Open}(s), E_V((s-t)u_1 + \text{“}\mathit{Open}(g^v)\text{”}), E_V((s-t)u_2 + \text{“}\mathit{Open}(g^r)\text{”})
\end{array}$$

Fig. 4. Protocol 3: schnorr identification with fair coin flip and blind decommitment

authentication trace and hence the correctness of the homomorphic computation. As a consequence we do not need verifiable encryption by the prover. The protocol proceeds as in Fig. 4.

Theorem 4. *Protocol 3 is* reverse unobservable.

Proof. The prover only receives one message $E_V(-t)$. The simulator simulates this message using a random ciphertext due to the semantic security of the encryption scheme.

If $z \neq 0$, then the terms $(s-t)u_1$ and $(s-t)u_2$ are independently uniformly random, since u_1 and u_2 are independently uniformly random. Hence, we can simulate these messages using randomly chosen ciphertexts and the commitments are never opened. The other messages are as in the proof of Theorem 2. Note that we employ a weaker notion of fairness. The verifier could force $z \neq 0$, since he could choose $t > \alpha$, but that does not seem rational in our scenario and is hence not included in the security definitions. As a consequence we do not need verifiable encryption by the verifier.

The prover's ciphertext can be checked for correctness after decryption, if $z = 0$, since it then needs to contain a correct authentication trace. If $z \neq 0$, the plaintext is random and must not be checkable. Since the verifier receives s and can check this against the commitment, he reliably knows z and can hence decide whether to check the plaintext. Consequently, also the prover does not need use verifiable encryption.

4.4 Optimization

There are a few tricks we can use to make the protocol more efficient. First, we translate our Protocol 3 to the elliptic-curve (EC) setting and employ the EC-ElGamal additively homomorphic encryption scheme [6]. This way, we can avoid computationally expensive modular exponentiation (e.g. required by the Paillier scheme) on the prover-side, which potentially uses a device with limited computation resource. Let $\mathcal{P}$ be a point on an elliptic curve over a finite field and

$$
\begin{aligned}
&P \rightarrow V : \mathrm{MAC}(s, K_1), \mathrm{MAC}(v\mathcal{P}, K_2), \mathrm{MAC}(r\mathcal{P}, K_3), r + H(r\mathcal{P})v \\
&V \rightarrow P : w\mathcal{P}, -t\mathcal{P} + w\mathcal{V} \\
&P \rightarrow V : s, K_1, u(w\mathcal{P} + y\mathcal{P}), u(-t\mathcal{P} + w\mathcal{V} + s\mathcal{P} + y\mathcal{V}) + \mathcal{K}_4, \mathrm{ENC}_{K_4}(v\mathcal{P}|K_2|r\mathcal{P}|K_3)
\end{aligned}
$$

Fig. 5. Protocol 4: optimized protocol 3 in the EC setting

that generates the required cyclic subgroup. The prover's public identifier is then translated into $\mathcal{Q} = v\mathcal{P}$ for some random v and g^r is represented as $r\mathcal{P}$. Hence, let $\mathcal{V} = x\mathcal{P}$ be the public encryption key of V, where x is the corresponding secret key, EC-ElGamal encryption of $-t$ is of the form $(w\mathcal{P}, -t\mathcal{P} + w\mathcal{V})$ for a randomly chosen w. In the last step of the protocol, P encrypts s in the form of $(y\mathcal{P}, s\mathcal{P} + y\mathcal{V})$ for a random y, homomorphically adds the ElGamal encryption of s to the encryption of $-t$ from V, and blinds the sum with u and adds an encoding $\mathcal{K}_4$ as a point on the elliptic curve of some random symmetric key K_4. This symmetric key is used to encrypt and protect the openings in a symmetric authenticated encryption mode like GCM. The complete protocol is illustrated in Fig. 5.

Moreover, we notice that $Commit(r\mathcal{P})$ does not need any random coins. It has sufficient entropy. Hence, $Open(r\mathcal{P})$ can be just $r\mathcal{P}$. Also, we do not need to encrypt s, but simply use a MAC that takes as input s and a freshly chosen random seed K.

We note that although EC-ElGamal decryption can be very slow (on the orders of seconds and minutes) even for small messages [12], such an operation is not required in our protocol. Given s (in the last step of the protocol), the verifier checks if $z = 0$. If so ($s = t$), V recovers $\mathcal{K}$ using its secret key v (to compute the term $uw\mathcal{V}+uy\mathcal{V}$). Otherwise, it simply does nothing and terminates the protocol.

4.5 Efficiency Analysis

We now analyze the computational and communication overhead of our EC-based protocol. We use a standard 160-bit elliptic curve `secp160r1`, which offers roughly the same security level as 1024-bit RSA. It is known that the dominant cost in EC-based cryptographic operations is point multiplications [10]. On the other hand, modular addition, modular multiplication, SHA-1 and AES operations are roughly three to four orders of magnitude faster than a point multiplication evaluation [31]. Hence, in our analysis, we consider only elliptic curve point multiplication (such as $r \cdot \mathcal{P}$) and elliptic curve double multiplication (such as $r \cdot \mathcal{P} + w \cdot \mathcal{V}$). Also, we believe that our analysis below represents a fairly conservative estimate for the amount of computational cost required by our protocol, since the actual processors used in real life would likely to be more powerful.

Table 1. Estimated computation time (in ms) for the prover and the verifier

	Offline		Online		Total	
Prover						
– Smart card	1075		654		1729	
– Smart phone	404		246		650	
Verifier	$(z \neq 0)$	$(z = 0)$	$(z \neq 0)$	$(z = 0)$	$(z \neq 0)$	$(z = 0)$
– Auditor	0.98	0.98	0	0.9	0.98	1.88

Prover. We consider two popular embedded microprocessors: 8-bit Atmel ATmega AVR and 32-bit ARM Cortex-M0+ microprocessors. The former represents a low-end contactless smart card (potentially used as an e-ticket in public transportation [17]), while the latter represents a low-end smart phone (potentially used as an OBU for toll collection [21]) in comparison with today's fast-evolving phone market with increasingly more powerful devices emerging.

According to recent implementations by Wenger et al. [32], the computation time required for a point multiplication is 327 ms with the 8-bit microprocessor and 123 ms with the 32-bit microprocessor; while a double multiplication takes roughly 421 ms and 158 ms, respectively. Their implementations are optimized based on set-instruction modification. The measurements are taken at a clock rate of 10 MHz. From these, we can estimate that the computation time required by the prover for one protocol run is roughly 1.7 s on a smart card and roughly 0.7 s on a smart phone. With (offline) pre-computation of some parameters used in the protocol, the computation time can be reduced by almost 60 % for both platforms. The computational cost of our protocol is summarized in Table 1.

In terms of communication cost, the prover sends two messages to the verifier during each protocol run. With our choice of elliptic curve, each message's length is only approximately $3 \times 160 = 480$ bits, and hence, the total bandwidth requirement is 960 bits per protocol run.

Verifier. We assume that the verifier is an auditor (local government), who will perform spot checks, and has much higher computational resources than the prover. A point multiplication and a double multiplication in the EC setting take 0.45 ms and 0.53 ms, respectively, using MIRACL compiled with GCC (with standard level-2 compiler optimization) and running on an Intel Single Core i5 520M 64-bit processor at 2.40 GHz. Given this, we estimate that in a protocol run, the computation time taken by the verifier is roughly 0.98 ms when $z \neq 0$, and 1.88 ms when $z = 0$.

On the other hand, the communication overhead incurred by the verifier is minimal. It sends out only one message of $2 \times 160 = 320$ bits.

In summary, our performance analysis shows that our protocol is very efficient using practical parameters. Using the toll collection scenario, assuming that the prover can pre-compute the necessary parameters before approaching a surveillance monitor[2], and without considering the communication latency between the verifier and the prover (via an OBU), our protocol can perform spot checks on up to roughly 4 vehicles in a second. We reiterate that our estimate is somewhat conservative. Newer ARM-based microprocessors and smart phones are likely to have even better computing speed.

Our protocol is designed to be lightweight. We omit verifying the recorded time and location by including a trustworthy GPS sensor and clock on the spot checking device. Hence, this information by the verifier is trusted. The verifier is not trusted to observe the privacy rights of the prover. Our protocol ensures this.

4.6 Rate Limiting

The frequency at which the spot checking identification device can be queried needs to be rate limited. Since we do not use any verifier authentication, anyone can query the device potentially tracking the user. If we assume a time τ between any two protocol initiations, we can estimate the average time t_{id} until a successful identification in a non-stop querying attack as

$$t_{id} = \frac{\alpha}{2}\tau$$

This delay τ needs to be traded against potential denial-of-service attacks. A driver (in the toll collection example) can be delayed until the spot checking protocol has been completed. Hence, a malicious reader can read the device just before a spot checking point and delay the driver and traffic. We estimate values between 5 and 30 s to be good choices for τ.

Another solution to this problem would be reader authentication. The reader could sign his message using a public key. The verification of the signature on the spot checking device would require at least one ECC multiplication for a fixed public key or at least two for public key certificates plus certificate handling. We opt against this design choice rather increasing the speed of our protocol.

4.7 Disposal

A simple idea of the prover to evade spot checking could be to remove or damage the spot checking device. Fortunately, this is very simple to detect: at any time, the verifier initiates a protocol, it is not completed by the prover. The security guarantees of our protocol design ensure that if the protocol is completed a valid public identifier is revealed with probability $1/\alpha$. Hence, a simple physical strategy can be employed to pursue provers without working spot checking devices. For example, a prover could not be allowed to proceed at a gate, photographed in flowing traffic as in many US toll collection sites or even chased by the police. Once caught, the police can even verify that the device is not working properly by re-running the protocol.

[2] The time required to travel from one spot to another spot, i.e. the distance between two surveillance monitors, would be abundant for our pre-computation purpose.

5 Example Application

For completeness, we give a sketch on how our privacy-preserving spot checking approach can be integrated with the Milo protocol for electronic toll collection [22].

We first describe a simplified version of the original Milo protocol[3] between the driver, toll service provider and the toll charger (local government):

Setup: The OBU generates the necessary key material and publishes the unique identifier id, including a signing key. The toll service provider and the toll charger each stores a copy of the OBU's verification key and its corresponding id. Hidden cameras are installed at random road segments and operated by the toll charger.

Payment Protocol: As the driver travels, the OBU picks up location ($where$) and time ($when$) information. The OBU then calculates the toll fare d for each road segment based on the received information. It also computes a commitment to the d value and encrypts the $(where, when, d)$ tuple.[4] At the end of each billing month, the OBU transmits the billing information, which comprises its id and a set of signed, encrypted and committed $(where, when, d)$ tuples, to the toll service provider. The latter verifies the committed toll fares via zero-knowledge proofs; if the check succeeds, it forwards the billing information to the toll charger.

Audit Protocol: For each captured vehicle via a hidden camera, the toll charger identifies the corresponding id and stores the $(id, where, when)$ tuple. At month's end and upon receiving an audit request from the toll service provider (with the relevant encrypted billing information submitted by the driver), the toll charger randomly selects n tuples where the corresponding id matches the identifier for the driver to be audited. It then requests for the decryption keys corresponding to the selected $(id, where, when)$ tuples in an oblivious manner (otherwise locations of the cameras would be revealed to the driver.) Upon decryption, the toll charger is convinced that the driver had behaved honestly only if the committed toll fares are correct for the checked road segments.

Using our privacy-preserving spot checking technique, some level of user privacy can be preserved,[5] while ensuring cheating drivers can be detected through the above described audit protocol. This requires the following small modification. Instead of using cameras, the surveillance monitors here are readers installed at gantries or booths setup along the roadside for all segments within an audit area. Moreover, we assume each vehicle uses an OBU to interact with the spot checking reader. Whenever the reader detects an approaching vehicle, it runs our proposed spot checking protocol. For each protocol run and if $z = 0$, the reader recovers and transmits the public identifier PK of the interacted OBU

[3] Many details of the Milo protocol have been omitted. See [22] for a complete description.

[4] Anonymous identity-based encryption is used here. The encryption key for each record is based on the $(where, when)$ tuple.

[5] We note that perfect privacy implies inability to detect dishonest users.

to a centralized server managed by the toll charger; otherwise, the reader sends nothing to the server. The rest of the Milo protocol remains the same.

There are two advantages for using our privacy-preserving spot checking mechanism. First, the toll charger provably sees only a fraction of the user's traveled road segments and spot checking is done randomly, i.e. the toll charger has no influence to which road segments it wants to check. That is our protocol actually preserves privacy and still discourages dishonest users from cheating by preventing any form of information sharing. Second, our approach potentially has lower operation cost. Particularly, the toll charger stores records associated with only a small fraction of the vehicles detected by the reader. Spot checking via cameras requires much higher storage and processing overhead. This is because each camera typically stores information of all vehicles that it captures.

6 Conclusions

In this paper we investigate collusion attacks on privacy-preserving billing or accounting systems. We show that in order to deter cheating all transactions in the cyber-physical world need to be observed or extreme penalties need to be imposed. This is not a sustainable choice for our society. We then present a privacy-preserving spot checking protocol that allows privacy-preserving observations in public spaces and can be used to enforce honesty in billing while still preserving privacy. For this we introduce a new variant of oblivious transfer: blind decommitment. We show that it can be efficiently implemented on weak computational devices such as smart cards or smart phones using a number of optimizations.

Our technique allows a socially acceptable trade-off between necessary observation (in public spaces) and privacy. We conclude that it is feasible to build cyber-physical billing systems that are economically dependable and privacy-preserving at the same time.

A Privacy vs. Penalty Analysis

We first analyze the probability of cheating detection with our spot checking mechanism. We then analyze how much privacy is achieved through our protocol and the required penalty to discourage users from cheating.

A.1 Variables

Let us assume that an average user would commute for a distance that covers k spots within a month. At each spot, only a fraction $\frac{1}{\alpha}$ of users are observed. Here we regard α as an "privacy indicator". The higher the value of α, the higher the level of user (location) privacy can be preserved. Let also $\mathsf{C}(k)$ denote the event when a cheating user is caught (or detected) *at least* once after having traveled k spots. (For simplicity, we ignore the cases where a cheating user is detected more than once within a month.)

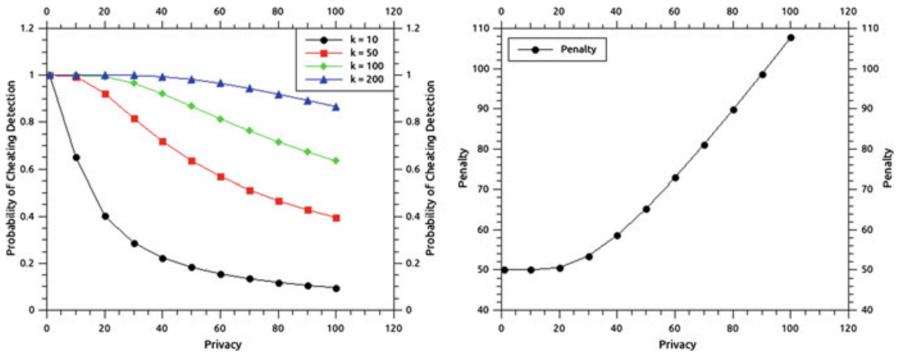

Fig. 6. (a) The relation between privacy parameter α and the probability $\Pr[\mathsf{C}(k)]$ of cheating detection at least once after k spots. (b) The relation between privacy paramater α and the imposed penalty p for any cheating user.

A.2 Analysis

Using a probability analysis similar to that of [2,22], we have

$$\Pr[\mathsf{C}(k)] = 1 - (1 - \frac{1}{\alpha})^k \tag{2}$$

for $\alpha \geq 1$ and $k \geq 1$. Clearly, when $\alpha = 1$, there is no privacy; however, the relevant surveillance monitor will detect any cheating user with probability 1. On the contrary, the more user privacy we want to preserve, the harder it is for the surveillance monitor to detect any cheating user. Clearly, users who have traveled more spots (higher mileage) have higher chances of getting caught if they cheat. Figure 6(a) shows that for a user with very low monthly mileage ($k = 10$), α needs to be set very low (hence low privacy) between the range of 5 and 10, such that any cheating can be detected with probability 0.8. On the other extreme, a user with high monthly mileage ($k = 200$) can still be detected with high probability (> 0.8) if cheating, while enjoying a higher-level of privacy ($\alpha = 100$). For an average user (say $k = 100$), α needs to be set at most 60 in order to have probability of cheating detection of at least 0.8.

We now quantify how much penalty p needs to be imposed for a cheating user to alleviate dishonest behavior. Let d be the toll fare for each traveled spot. From Eq. (1), we have

$$p(\Pr[\mathsf{C}(k)]) - dk(1 - \Pr[\mathsf{C}(k)]) \geq \epsilon$$

where ϵ is the minimal net loss[6] that will incentivize a user to behave honestly. (Clearly, if the net loss is close to zero, there is little incentive for the user not

[6] Here net loss is assumed to be the difference between a fine (penalty) for being caught cheating and the amount of money that a cheating user would have saved should her dishonest behavior was not detected.

to cheat.) Setting $d = 0.50$ USD, $\epsilon = 50$ USD and $k = 100$, we can then define the relation between p and α as

$$\begin{aligned} p &= \frac{\epsilon + dk(1 - \Pr[\mathsf{C}(k)])}{\Pr[\mathsf{C}(k)]} \\ &= \frac{50 + 0.5(100)(1 - \Pr[\mathsf{C}(k)])}{\Pr[\mathsf{C}(k)]} \\ &= 50\left(\frac{1 + (1 - \frac{1}{\alpha})^{100}}{1 - (1 - \frac{1}{\alpha})^{100}}\right) \end{aligned} \qquad (3)$$

It is clear from Eq. (3) that p increases when α grows. Figure 6(b) shows that for the case of an average user (say $k = 100$), p grows at a lower rate for $\alpha < 30$, but at a much higher rate for $\alpha > 30$. To achieve a reasonable level of privacy while ensuring honesty for an average traveler (following our earlier probability analysis that infers $\alpha \leq 60$ and $\Pr[\mathsf{C}(k)] \geq 0.8$ for $k = 100$), we must impose penalty of approximately \$70 for users who cheat at least once within a month.

We would like to point out how difficult it is to achieve a similar level of privacy using mobile cameras. One can think of the following hypothetical mobile camera. If we assume that a mobile camera (mounted on a special-purpose vehicle) moves in an unpredictable direction at a speed much higher than traffic and may stay at a specific spot for an unpredictably small amount of time, it is roughly saying that the camera is installed (fixed) at a location for a small fraction of time. Still, the mobile camera records a small fraction of users, but likely more than one as in our case who appear at the location where the camera is. Moreover, as analyzed by Meiklejohn et al. [22], the operational cost of even existing mobile cameras is much higher than that of fixed cameras. Their analysis shows that an audit vehicle costs at least 82,500 US dollars per year. This includes the cost for employing a driver, as well as purchasing, operating and maintaining the audit vehicle. Our approach of spot checking offers more economical and fine-grained control (only one is observed) of the levels of cheating detection and user privacy by adjusting the relevant parameters.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 119. Springer, Heidelberg (2001)
2. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: privacy-preserving electronic toll pricing. In: Proceedings of the 19th USENIX Security Symposium (2010)
3. Brassard, G., Crépeau, C., Robert, J.M.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
4. Catrina, O., Kerschbaum, F.: Fostering the uptake of secure multiparty computation in e-commerce. In: Proceedings of the International Workshop on Frontiers in Availability, Reliability and Security (FARES) (2008)

5. Dreier, J., Kerschbaum, F.: Practical privacy-preserving multiparty linear programming based on problem transformation. In: Proceedings of the 3rd IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT) (2011)
6. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
7. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM **28**(6), 637–647 (1985)
8. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
9. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
10. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004)
11. Heydt-Benjamin, T.S., Chae, H.-J., Defend, B., Fu, K.: Privacy for public transportation. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 1–19. Springer, Heidelberg (2006)
12. Y. Hu.: Improving the efficiency of homomorphic encryption schemes. Ph.D thesis, Worcester Polytechnic Institute (2013)
13. Kerschbaum, F.: Building a privacy-preserving benchmarking enterprise system. Enterp. Inf. Syst. **2**(4), 421–441 (2008)
14. Kerschbaum, F.: A verifiable, centralized, coercion-free reputation system. In: Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society (WPES) (2009)
15. Kerschbaum, F.: Outsourced private set intersection using homomorphic encryption. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communication Security (ASIACCS) (2012)
16. Kerschbaum, F., Dahlmeier, D., Schrpfer, A., Biswas, D.: On the practical importance of communication complexity for secure multi-party computation protocols. In: Proceedings of the 24th ACM Symposium on Applied Computing (SAC) (2009)
17. Kerschbaum, F., Lim, H.W., Gudymenko, I.: Privacy-preserving billing for e-ticketing systems in public transportation. In: Proceedings of the 12th Annual ACM Workshop on Privacy in the Electronic Society (WPES) (2013)
18. Kerschbaum, F., Terzidis, O.: Filtering for private collaborative benchmarking. In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 409–422. Springer, Heidelberg (2006)
19. Kerschbaum, F., Oertel, N.: Privacy-preserving pattern matching for anomaly detection in RFID anti-counterfeiting. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 124–137. Springer, Heidelberg (2010)
20. Kilian, J.: Founding crytpography on oblivious transfer. In: Proceedings of the 20th ACM Symposium on Theory of Computing (STOC) (1988)
21. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Enhancing location privacy for electric vehicles (at the *Right* time). In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 397–414. Springer, Heidelberg (2012)
22. Meiklejohn, S., Mowery, K., Checkoway, S., Shacham, H.: The phantom tollbooth: privacy-preserving electronic toll collection in the presence of driver collusion. In: Proceedings of the 20th USENIX Security Symposium (2011)

23. MIRACL - Benchmarks and Subs. Certivox Developer Community (2014). https://certivox.org/display/EXT/Benchmarks+and+Subs
24. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
25. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
26. Popa, R.A., Balakrishnan, H., Blumberg, A.J.: VPriv: protecting privacy in location-based vehicular services. In: Proceedings of the 18th USENIX Security Symposium (2009)
27. Rabin, M.: How to exchange secrets by oblivious transfer. Technical Memo TR-81, Aiken Computation Laboratory (1981)
28. Sadeghi, A., Visconti, I., Wachsmann, C.: User privacy in transport systems based on RFID e-tickets. In: Proceedings of the 1st International Workshop on Privacy in Location-Based Applications (PilBA) (2008)
29. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
30. Schröpfer, A., Kerschbaum, F., Müller, G.: L1-an intermediate language for mixed-protocol secure computation. In: Proceedings of the 35th IEEE Computer Software and Applications Conference (COMPSAC) (2011)
31. Uhsadel, L., Poschmann, A., Paar, C.: Enabling full-size public-key algorithms on 8-bit sensor nodes. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 73–86. Springer, Heidelberg (2007)
32. Wenger, E., Unterluggauer, T., Werner, M.: 8/16/32 Shades of elliptic curve cryptography on embedded processors. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 244–261. Springer, Heidelberg (2013)