A (preliminary) note on privacy friendly public transport ticketing*

Jaap-Henk Hoepman

In this note we ask ourselves whether it is possible to design a privacy friendly system to pay for public transportation.

We study three possible approaches, and explain how they work in an easy to understand manner, and analyse their properties and discuss their strengths and weaknesses.

This note is for a large part based on our experiences with the Dutch OV chipcard system, and recent proposals for updates to that system [4]. The trigger to write it is a recent court case concerning the lack of privacy of this nationwide system for paying for public transportation, and especially the fact that the so-called anonymous OV chipcard is really not anonymous.

1 Preliminaries

1.1 Stakeholders and system model

We assume a system that supports many modes of transportation. This means we distinguish several *public transport operators (PTOs)* that offer public transport services. *Users* travelling by public transport make trips that may consist of several legs each using a different mode of transportation offered by a different PTO. A *central clearinghouse* provides the public transport ticketing infrastructure (or at least the APIs to connect to this infrastructure), handles payments from *banks* initiated by users, and distributes financial compensation to the PTOs for services rendered.

Users have a digital *token* that enables them to travel by public transport. Instead of relying on a smart card (to store tickets or other information needed to verify whether someone is entitled to a certain mode of public transportation) we may also assume most people own a sufficiently modern and capable smartphone, and are willing to use it for public transport (even though in privacy terms an ill configured smartphone is like having a never sleeping Stasi agent in your pocket).

* Thu Sep 5 14:28:35 2019 +0200 / ae529f1 / ov-pet-note.md

1.2 Requirements:

The system should satisfy the following loosely formulated requirements.

- Users should pay for trips, where the fare depends on when and where one travels, as well as on the distance travelled, and where users can subscribe to receiving reduced fares (during certain times of the day, or on certain tracks).
- Public transport operators should receive compensation for their services, which (partly) depends on all the actual trips that used part of their infrastructure. In other words, the amount of compensation can depend on how many passengers travelled on which particular track on which day.
- The system should be privacy friendly: no party should be able to link any number of trips to each other (as belonging to one, unknown, person) or to any one particular person. In other words, the previous requirement only allows the PTO to learn *how many* passengers travelled a certain track at a certain time of the day, not *who* they were, or whether they were the *same people* that set out on some other trip earlier.
- The system should be secure: it should prevent or detect fraud by users (e.g. fake tickets, pay less than the required fare) and prevent fraud by operators (e.g. claiming more trips than actually took place over their infrastructure).
- The system should be fast enough to process large volumes of travellers at peak hours. Checking travellers by conductors should take less than a second. Checking in or out to enter or exit public transport (like in the Dutch OV chipcard system, the London Oystercard system, or many other metro and bus systems in the world) should take only a few hundred milliseconds at most.
- The system should be user friendly (whatever that may mean. . .).

1.3 Threat model

We assume users are willing to actively defraud the system (travelling by public transport by paying less than required or nothing at all), if the probability of being caught is low. They will buy fraudulent smart cards or root their smartphones and install fraudulent apps if there is a clear benefit. This means that in terms of smartphones we cannot assume any trusted environment to store secrets, in other words the device and the app are untrusted from the PTO perspective. For smart cards we

assume that they *can* be used to store secrets and keep them confidential, preventing their users from accessing them.

We assume banks, PTOs and the central clearinghouse will actively (and collectively) try to break privacy and recover trip details from their users, using any information they can get their hands on. They are untrusted from the user perspective.

We do assume however that PTOs do *not* try to break privacy by writing their apps in such a way that the information provided by the user through the app, but shielded from the central PTO servers by the protocol, is surreptitiously sent to the PTOs regardless. The PTOs could in theory do this. We can mitigate this by offering third party (open source) apps, requiring external audits and analysis, or through the vetting procedures enforced by the smart phone app stores. (This is another reason why we cannot assume that the smartphone or app can store secrets.)

We assume that PTOs will try to defraud the system and claim more compensation from the central clearinghouse than warranted. The clearinghouse is trusted, in the sense that it does not favour one PTO over the other, and that at the of the day all money received must be spent (on compensating PTOs or the cost of running the clearinghouse and its ticketing infrastructure). Audits can be used to ensure this.

We assume the cryptographic primitives used cannot be broken, and that entities keep their secrets secret (unless they could benefit from not doing so).

1.4 Other assumptions

We assume secure, i.e. authenticated and encrypted, connections with all banks, public transport operators and the clearinghouse. Clearly the user is not authenticated (although the token of the user may be verified to be authentic).

We assume fares are course enough to ensure that the price associated with a trip does not reveal the actual trip itself. For example, trip prices could be set at fixed amounts for every ten kilometres travelled, with a fixed ceiling fare for all trips longer than a certain distance. (Care should be taken to ensure that for every possible fare the number of different trips with that fare is sufficiently high to guarantee a reasonable degree of anonymity.)

We also assume that local, immediate device to device, communication is using only ephemeral identifiers (if any) to prevent linking devices over longer periods of time. This means WiFi or Bluetooth are using properly randomised MAC addresses, or random anti-collision identifiers if NFC is used. This also implies that we assume apps do not have access to any other permanent, unique, device specific identifier[1].

For normal (long range) internet connections between the smartphone and the servers of the public transport operator or the bank we cannot make such an assumption: it is rather trivial to track users based on their often fixed IP addresses. [2].

The tacit assumption in this work is that it is a lot safer, from a privacy perspective, to collect personal data locally on the user device, instead of centrally on the servers of the service providers. Clearly a malicious public transport app can collect and upload all this personal data surreptitiously anyway. The assumption (see above) is that this cannot or will not happen.

There are many practical constraints that limit what we would ideally like to do. For instance, there already is a more or less standardised way to initiate and handle payments from bank accounts (called iDeal in the Netherlands), that has to be used but is unfortunately not very privacy friendly: in the end the payment intermediary or the payee (in our case the PTO or the clearinghouse) receives the transaction details, including the bank account details of the payer (in our case the user).

[1] The operating systems of these smartphones should, could and sometimes actually do prevent apps from having access to such a persistent identifier. Preventing the app itself to generate such an identifier itself and store it locally is of course not possible (although audits may reveal this).

[2] Users could mitigate this risk using a Virtual Private Network (VPN): in the case the PTO servers only see the IP address of the VPN provider, and would need the cooperation of the VPN provider to obtain the IP address of the user. This makes tracking or tracing users a lot harder (but certainly not impossible). One *meta conclusion* of this work is that we need an efficient, frictionless, way to provide sender anonymity on the Internet, similar to the use of randomised MAC addresses that can be used on local networks. A VPN is too weak (the VPN provider sees everything its users do), yet Tor is too strong (there is no need to protect against a NSA like adversary) given the impact on performance. If randomised client IP addresses could be used by default to set up a TCP connection between a client and a server, that would already provide a tremendous boost in privacy on the Internet as servers can no longer trace their users based on their IP address.

1.5 Core technology 1: partially blind signatures

One of the core problems in trying to make public transport ticketing privacy friendly is the need to provide a clear separation between the actual payment (which, when involving a bank account, is necessarily identifying), the payment receipt (from which this identifying information should be stripped), or the transportation ticket (that contains the trip details).

Taking a transportation ticket as an example, we need something that allows us to securely issue tickets anonymously, which can then be used only once by a user, and can subsequently be redeemed by the public transport operator (see also [7]).

We use *partially blind signatures* as the core technology to implement such tickets. A blind signature allows a user to obtain a signature on a message from a signer, without the signer actually learning the message being signed. Moreover, the signed message cannot be traced back to when it was signed: the issuer cannot correlate the signatures it created with signatures later provided for verification. In other words, the signature also needs to be *untraceable*. Essentially, the user transforms the signature after issuing to avoid such linking. Blind signatures were invented by David Chaum [3], who also provided the following nice metaphor. A blind signature is like sending a message inside a sealed envelope to the signer, where the inside of the envelope is covered with carbon paper. This means that if the signer stamps (with a fixed stamp) the envelope from the outside with its signature, the carbon paper transfers this signature to the secret message inside the envelope. When the signer returns the still sealed envelope (proving it didn't see the message) all the user needs to do is to open the envelope to obtain the blindly signed message.

A partially blind signature allows the signer to add some known information to the message. In other words: part of the message to be signed is secret and provided by the user. The other part of the message to be signed is known and provided by the signer. After the blind signature protocol, the user has a valid signature from the signer on the whole message consisting of both the secret user-provided attributes[3] and the known signer-provided attributes (and these cannot be separated, nor can any of the attributes be changed without invalidating the signature).

[3] An attribute is an arbitrary piece of information, typically a piece of personal data, like your date of birth, or your current location.

Efficient partially blind signatures exist [1, 5].

1.6 Core technology 2: attribute based credentials

We use an attribute based credential (ABC) scheme, where users can request credentials containing attributes from issuers, and can show a subset of these attributes to relying parties at a later time.

Credentials are the *secure* containers of these attribute, meaning that users cannot create valid credentials themselves, nor can they change the contents and in particular the attribute values contained in these credentials. This ensures that relying parties can indeed rely on the validity of the attribute values being shown to them.

ABCs are also *privacy friendly*. Multiple showings of attributes in a credential are *unlinkable* to their originating credential (provided the attributes themselves are not uniquely identifying or linkable). This in particular prevents the issuer to link subsequent showings of the credentials it issued. Moreover, users can choose to *selective disclose* only a subset of all attributes contained in a credential, hiding the values of the undisclosed attributes to the relying party.

Often, attribute based credentials are based on zero knowledge proofs. The trick being that the user proves in zero knowledge that it has a credential with valid signature and the necessary attributes, without actually revealing the credential itself. This by definition guarantees unlinkability. Interactive zero knowledge proofs have the additional property that they are deniable, meaning that the verifier cannot convince someone else that a particular prover proved a statement. Non-interactive zero knowledge proofs (typically used for ABCs) do not have this property and are undeniable [6]. For many applications this is a good thing, as it allows the logs of the verifiers to be audited.

The ABC scheme we use here should have the following additional properties.

It should be possible to request a credential where some of the attribute values are set by the user and hidden from the issuer. In other words, the issuer must be able to sign a credential without learning the attribute values of some of the attributes, without allowing the user to change these attribute values afterwards. These attributes are called *blinded*.

Certain credentials can be marked for *one-time use* at time of issuing. This means the credential can only be shown once at a relying party.

After that the credential becomes invalid and any subsequent showings will fail. One time use can be achieved, for example, by adding a random sequence number to a credential as a blindly issued attribute (ensuring issuer unlinkablity), which must be shown at all times and becomes blacklisted the first time it is used. Partially blind signatures can also be used as one time credentials, by the way (and are also undeniable, like non-interactive zero knowledge based credentials). In what follows we use partially blind signatures instead of full blown one-time use credentials, as we believe these to be more efficient.

Idemix [8] is an ABC scheme that satisfies all of our requirements (and is based on non-interactive zero knowledge proofs and therefore undeniable).

2 Solution 1: Emulating paper tickets

Of course one way to achieve privacy in public transport ticketing is to emulate the traditional use of paper public transport tickets. The basic idea is to first buy the ticket online, and subsequently use it for public transport later, in such a way that the account used to pay for the ticket cannot be linked to the actual trip being made.

2.1 Detailed protocol

We assume the user has a smartphone with a public transport app installed. Below we discuss variations to the protocol that allow a smart card to be used. We furthermore assume the app contains a database with all possible trips that can be made by public transportation, and the corresponding prices to be paid.

The protocol to obtain a ticket now runs as follows.

- The user selects the trip she wishes to make, and the date she wishes to make the trip.
- The app calculates the fare for the trip and then redirects the user to the payment app (associated with her bank account) to start payment of this fare.
- If the payment is successful, the app receives a payment receipt (which contains the amount paid) signed by the bank as confirmation

of this. This receipt should not contain identifying information[4], but should contain a sequence number to prevent reuse. The paid fare is collected by a clearinghouse that later redistributes the paid fares to the PTO based on submitted receipts.
- The app sends this receipt (which contains the price paid) to the public transport operator to request the issuing of the ticket. The ticket is issued using a partially blind signature. When requesting the ticket to be issued the actual trip details (trip start, trip end and date) as well as a random serial number are requested as blind attributes (to ensure the PTO does not learn these when issuing the ticket). This prevents the PTO from linking actual trip details to the payment receipt or with a device identifier or the IP address of the device. The fourth attribute in the ticket is the (unblinded) price paid for the ticket, added by the PTO based on the payment receipt. So a ticket is a one-time use credential containing the following four attributes: the three blind (hidden) attributes start, end and data, as well as the (known) price.

Public transport operators need to verify that all users that travel with them have a valid ticket. The conductor uses a smartphone for that purpose. The user sends the ticket (revealing all attributes, i.e. trip start, trip end, date, serial number, and price) with its (partially blind) signature. The smartphone of the conductor verifies whether the trip requires the user to be in this train/metro/bus/tram/etc., whether the price corresponds to the trip, and whether today corresponds to the ticket date. It also invalidates the signature, so that it cannot be used again[5].

The PTO is reimbursed based in the payment receipts it submits to the clearinghouse.

[4] Relatively little harm is done when the receipt does contain identifiable information, as the ticket is issued blindly anyway, and we more or less assume the clearinghouse, PTOs and banks collude.

[5] This means that the random serial number must be sent by the conductor to the PTO back office. Conductors must therefore be online, also to get updated about invalided credentials by other conductors. As tickets are only valid for a single day, PTOs may choose to forfeit on this strict form of checking (hence relaxing system requirements), relying on the fact that ticket can still only be used (perhaps multiple times) on a single day.

2.2 Extensions

Instead of relying on users having smartphones, tickets could actually be printed on paper[6], or be stored on smart cards instead. In this case, a ticket kiosk needs to be used to allow users to select the ticket they need, allow them to pay (by cash or card), and to print the ticket or issue the ticket to the smart card. In the latter case, conductors need to carry NFC enabled smartphones that allow them to scan the smart card and read the ticket (with its signature) from the smart card. This is certainly possible with current (high-end) smart cards [REF].

2.3 Analysis

To what extent does this solution fit the requirements set out above?

Users obviously pay for their trips, and the fare depends on the distance travelled. Conductors and sufficiently high fines are necessary to keep users honest and disincentivise travelling without a valid ticket.

Public transport operators get paid based on the payment receipts they collect when issuing tickets. To get (statistical) information about actual trips made they need to have enough conductors to check the tickets of all their passengers when travelling (as this is the only time when the actual trip details are revealed). If multiple PTOs are involved in a particular trip, proper reimbursement can only be achieved if the app splits up the trip in different legs, one for each PTO the user needs to travel with.

The level of privacy protection is quite good but not perfect. In the setup described, the PTOs, clearinghouse and the bank could learn how many tickets you buy, and for which amount (ie for which distance), if it would try to identify you based on your IP address or device identifier[7]. Even if PTOs and banks collude, they would not be able to link bank account holders with actual trips made, but timing analysis linking payment times with ticket issuing times could be used by the PTO to be more certain about your identity. If you usually buy your tickets on the same day or

[6] This may sound silly or even childish, but in fact when trying to emulate something and doing something digitally analogous to how it was done physically, one always has to consider the option that the original, physical, approach simply works better.

[7] Note however that we assume that the device operating system prevents the app from having access to such a unique and persistent identifier.

the day before your trip, your PTO could learn when you travel. The PTO could learn whether you are using public transport a lot, or not[8]. Many short trips on the same day may reveal you are in a city; certain patterns of distances may correspond to popular tourist routes (and hence reveal the city you are in). This limited level of privacy protection may already be a threat for people that engage in civil disobedience, like the Hong Kong protesters or the Extinction Rebellion activists. All these problems can be avoided if users can use cash to buy tickets, at special digital kiosks.

Communication between the user and conductor smartphone uses ephemeral identifiers. This means ticket inspection reveals no personal information: the zero knowledge proof only reveals the ticket price and the trip details, but no personal identifier.

If we can guarantee that there is no way for remote servers to identify the devices that connect to them (because persistent device specific identifiers are blocked by the device operating system, and because users are told to use a VPN for additional privacy protection), then there is in no longer a need to blind the trip details from the PTO when requesting a ticket to be issued. However, this would require an additional change to the protocol to ensure that the receipt send to the PTO as proof of payment cannot later be used to retrieve the personal details of the bank account holder through which the ticket was paid (when PTO and banks decide to collude). In fact, a partially blind signature could be used for this purpose where the user provides a blind random sequence number and the bank provides the known amount paid. This implies a (major) change to the payment API (which we deemed impractical above). But it would allow the PTO to obtain perfect information about all trips people set out on, while still keeping these people anonymous.

The system is secure: tickets are only issued by the PTO when given a payment receipt for a certain amount, signed by a bank. Only banks can create such a signed proof of payment. The amount paid for a ticket is checked by the conductor when inspecting a ticket. Only PTOs can create a valid ticket (signed in partially blind fashion). This signature is also checked by the conductor. Finally the conductor checks whether the ticket entitles a person to travel when and where the conductor inspected her ticket. Failure of one of these tests means the ticket is invalid. The sequence number of the ticket (embedded to guarantee one-time use)

[8] If the bank issues the tickets instead, the bank could collect this information and would for sure know it is you.

is checked in real-time with an online database of sequence numbers of already inspected tickets. If the sequence number is already in the database, the ticket is invalid. Otherwise, the sequence number is added to the database.

The system is also secure in terms of preventing PTO fraud: PTOs are reimbursed based on signed payment receipts for actual amounts paid by the users. Only banks can sign these receipts. Collected tickets (that include the fare paid) together with their valid signatures can be used to cross validate this.

The only relatively time critical step is the inspection of the tickets by the conductor. This involves a simple signature verification step which should not consume a lot of time (if both conductor and traveller have sufficiently modern smartphones).

The system is not really easy to use, as it forces users to buy tickets in advance.

2.4 How to deal with failures?

Dealing with failures is always a challenge, but this is particularly the case in privacy friendly protocols where often the link between a user and her actions is deliberately broken. This means extra care needs to be taken to create some evidence that allows an entity to challenge a failure, while not eroding the privacy of the users. Below we describe some possible failures, and how they could be dealt with. See also [7] for additional measures that can be taken.

The user pays, but does not receive a payment receipt If banks keep a copy of the payment receipt, they can always reissue it on request. As a payment receipt can only ever be used once, there is no harm in giving it twice to a user.

The user wants to cancel a payment The user can return the payment receipt (which contains a unique sequence number) to the bank to rewind the transaction. The bank then forwards the payment receipt to all PTOs signalling not to accept this payment receipt when a user requests a ticket to be issued. Also, the transfer of money from the bank to the PTO will be reversed.

The user submits a payment receipt, but does not receive a ticket If the PTO keeps a copy of the last message it sent in the ticket issuing protocol, tied to the payment receipt against which it was issued, the user can request the PTO to resend this message (and

thus complete the issuing protocol to guarantee she receives the ticket). This should only be done, of course, if she can hand over the correct payment receipt. Again having two exact copies of the same blindly signed ticket offers no advantage to the user: she can only use one of the copies. If the issuing protocol fails earlier, the PTO is sure that no ticket was issued at all, so in that case the payment receipt is considered unclaimed and can be used to request issuing a completely new ticket.

The user submits a payment receipt, but receives an valid but incorrect ticket This can happen if the user entered the wrong trip details, or if some internal error caused the wrong ticket to be issued. The user can 'return' the ticket to the PTO, essentially running the showing protocol normally run when a conductor inspects the ticket. This invalidates the ticket. Using the same payment receipt she can start restart the issuing step, now with the correct trip details (assuming the fare is the same).

The user wishes to cancel a ticket issued to her After 'returning' the ticket the PTO as described in the previous case, she can then proceed to cancel the payment to the bank.

The user submits a payment receipt, but receives an invalid ticket This is more tricky. Ideally the issuing protocol should guarantee that a valid ticket is issued. If this is impossible, at least the issuer should somehow be able to tell, from the logs, that the user indeed did not receive a valid ticket. Otherwise bogus claims for invalid tickets could be submitted. This all very much depends on the particular issuing protocol used.

A valid ticket fails conductor inspection Ideally this should not happen. However, the user or conductor device may malfunction, and the communication between the two devices may be erroneous. If the ticket is valid, and the user app operates correctly, the user should at some point be able to convince the PTO she had a valid ticket when travelling.

The app crashes or malfunctions Reinstalling the app should be possible without loosing any stored tickets, or turning them invalid.

The user looses or deletes a ticket There is no way to recover from this situation. (Loosing a ticket could happen when inadvertently deleting the whole app together with all its data.)

We note that it is theoretically possible to create a kind of trapdoor within the tickets. Suppose the user would use a secret root value to derive, in a deterministic but irreversible manner (using a cryptographic

hash function for example), the random sequence number that makes the ticket one-time use. Then the user can at a later time prove it was her that used a certain ticket. This may be useful to resolve disputes. But such a construction also carries a grave privacy risk as this secret value can be used to undo all privacy protection in the system.

3 Solution 2: Pay as you go, with credit on device

Another approach to achieving privacy in public transport ticketing is to pay as you go, where you have some stored value or credit securely kept on your device to pay for your trips. This aims to emulate the traditional way of paying for (short) public transportation trips by cash with a few coins.

Such so called *stored value card* systems already exist in fact, e.g. OV chipcard in the Netherlands, the Oyster card in London and the Octopus card in Hong Kong. They record the credit as a simple counter value, and therefore require the use of a smart card to protect this counter value against tampering by the user[9]. The system is similar to (and actually arose from) payphone cards (that were popular in the late twentieth century). To further prevent or detect fraud, these existing schemes use a fixed unique identifier in practice, and copy the balance on the card as well as information about past trips to a central server. This means that such systems are never anonymous but pseudonymous at best, and the real identity of the user can easily be attached to the fixed identifier through, for example, a bank transaction used to top up the credit on the card, or a customer support inquiry whenever something goes wrong.

To turn this idea into a privacy friendly architecture, the credit on the device needs to be stored in such a way that the device itself no longer

[9] Because the device essentially needs to store something that has a significant monetary value — a credit maximum of 100 euro is not unreasonable for a national public transport system if you want people to travel several times without having to top up after every trip — it needs to be protected from malicious users. This means a smartphone is not the ideal device to use here, as it typically has few defences against tampering by its owner. Smart cards, that have such tamper proof or tamper evident features, fare much better in context where the system provider needs to be protected against malicious users. Smart cards were also used for payphone cards, are used as SIM cards for mobile phones, and have even been used for a national smart card payment system in the Netherlands: Chipknip and Chipper (which both sadly got scrapped several years ago).

needs to be trusted. This obviates the need for tamper proof or tamper evidence features, and more importantly scraps the need for fixed identifiers and additional centralised bookkeeping.

3.1 Detailed protocol

Again we assume the user has a smartphone with a public transport app installed, but a smart card could be used just as well.

We furthermore assume that there is a way to *check in* (i.e to record the location where you start your trip) and to *check out* (i.e. to record the location where you end your trip). Traditionally this is done with turnstiles or poles on the platform that need to be tapped with the public transport card. Smartphones with NFC can also be used similarly[10]. When you check in, the system verifies that you have enough credit to make a trip. The actual fare is calculated and automatically deducted from your device as soon as you check out.

Credit on the device is represented by a signed token (c, s), issued by a central clearinghouse using a partially blind signature scheme, containing the credit c as the known value, and a random serial number s generated by the device as the blindly signed value.

Users first load some credit on the device using the following protocol.

- The public transport app redirects the user to the payment app (associated with her bank account) to start payment of the amount she wishes to add to her credit.
- If the payment is successful, the app receives a payment receipt (which contains the amount paid) signed by the bank as confirmation of this. This receipt should not contain identifying information, but should contain a sequence number to prevent reuse. The paid fare is

[10] When smartphones with GPS are used, a location fence (for known train stations for example) can trigger a potential check in or check out that needs to be confirmed on the phone. Such a 'be-in be-out' approach is currently being considered in the Netherlands. But as the management of credit in this case involves a long range Internet connection between the smartphone of the user and servers of the PTO, privacy can no longer be guaranteed (unless the user uses a VPN). Alternatively, Bluetooth beacons can be used instead to reliably determine presence at a known public transport location. These can also be used as the first, anonymous, hop in the connection with the necessary backend infrastructure to issue tickets.

collected by a clearinghouse that later redistributes the paid fares to the PTO based on the actual trips being made.
- The app sends this receipt (which contains the price paid) to the clearinghouse to request the issuing of the credit token with credit c.
- The user device generates a random s and joins clearinghouse server in the joint protocol for the partial blind signature on the new credit token (c, s), hiding s from the clearinghouse.
- The user device stores this credit token.

If the user already has some credit on the device, then the following additional steps need to be taken right after the payment step.

- The users sends the credit token to the clearinghouse server
- The clearinghouse server verifies the signatures on the token, and verifies that s is fresh (not used or spent before)
- The clearinghouse server adds the current credit to the paid amount and is ready to issue a new credit token for the new credit c (supported for this by the clearinghouse server) .

The protocol to check in (between a user device and a local check-in device) then runs as follows.

- The user sends the credit token to the check-in device
- The check in device verifies the signature on the token, checks that the stored value is larger than some minimum credit required, and checks that the serial number s is fresh. If that is the case it sends the serial number s to the central clearinghouse server to mark it as used, so that it can not be used again to check in.
- The check in device returns a check in token (signed by the PTO) which contains the current location, time, and the serial number s in the credit token.

The protocol to check out (between a user device and a local check-out device) then runs as follows.

- The users sends the check in token and the credit token to the check out device.
- The check out device verifies the signatures on both tokens, validates the time on the check in token, and checks that s is also included in the check in token (no freshness is checked here).
- Based on the check in location and the current location the check out device (in cooperation with the central clearinghouse) computes the fare f and is ready to issue a new credit token for the credit $c' = c - f$.

- The user device generates a new random s' and joins the check out device (helped by the clearinghouse) in the joint protocol for the partial blind signature on the new credit token (c', s'), hiding s' from the clearinghouse. The clearinghouse creates the signature, and keeps a record of the deducted fare f for reimbursement purposes. The check out device also creates a check out token with its signature, that contains the current location, time, and the sequence number s in the credit token.
- The user device stores this new credit token. The user device also logs the check in and check out tokens (with PTO signature) in a local trip history (that can be consulted to resolve disputes).

The conductor needs to verify that every person travelling has a valid check in token on their device that corresponds to their current location and mode of transport. For that purpose the conductor device essentially runs the same protocol as the check out device, except for the issuing a new token step.

The PTO logs all check in and credit tokens submitted during check out, together with the check out location, to claim benefits for services offered. These claims are verified against the record of deducted fares maintained by the central clearinghouse.

3.2 Analysis

Does this solution fit the requirements mentioned in the introduction?

Provided the minimum credit necessary to check in exceeds the maximum possible fare (for the longest possible trip) when checking out, the protocol guarantees that users pay for their trips. Once credit drops below zero, users can no longer check in. Conductors and sufficiently high fines again are necessary to keep users honest and disincentivise travelling without checking in. They are also necessary when untrusted, dishonest, devices fail to behave correctly when checking out.

If it can somehow be physically guaranteed (using turnstiles for example) that users must check in to enter public transport, and must check out to leave public transport, conductors are no longer necessary (although they are of course still of paramount importance to guarantee safety and well being on the public transport).

Public transport operators get paid based on their own check in and check out logs, crosschecked with the fare deduction logs collected

by the central clearinghouse when it issues updated credit tokens for users that check out. This ensures that public transport operators get compensated for their efforts, but cannot claim more than they are entitled to.

The level of privacy offered is probably better compared to the previous solution. Payment is the only step that involves a fixed identifier (through the bank account). If you could top up your credit with cash using public transport ticket vending machines, even that form of tracking disappears. If you top up the credit on your device with larger amounts, there is much less correlation between the times you top up credit and the time you travel by public transport. PTOs learn less, and are no longer able to track when and how often you travel (through the ticket buying patterns revealed in solution 1). The banks still learn how much you travel, in terms of the amount of money you spend on public transportation, unless you use cash. Only check ins and check outs are linked (through the same serial number in the credit token). Fresh credit tokens are issued with new random serial numbers, that are kept secret when the credit token is issued. This breaks the link between past and current credit tokens. Unfortunately there is one issue that complicates the analysis: the actual credit on your token stays the same between check out and check in. This allows the PTO to link a check in with a particular credit to an earlier check out with the same credit. Now the number of possible credit values is not so large (10.000 if we assume a credit between 0 and 100 euro, measured in cents), and the number of people travelling large (in the order of millions). Moreover, a certain credit value at check out may occur many times over a larger period of time, and there is no way to tell for a PTO which of these previous check outs corresponds to the current check in with that same credit value. We suspect therefore that the privacy impact is insignificant.

A pay as you go approach with credit on the card has the potential of significant improvements in terms of usability as well. The fact that you no longer need to buy a ticket beforehand is the major benefit of this approach. However, it does require users to explicitly check in and check out for each and every trip (which takes a lot of getting used to), and to have sufficient credit on their tokens. Errors should be easy to fix or graciously dealt with[11].

[11] This is absolutely not the case with the OV chipcard system in the Netherlands where forgetting to check out means you pay the maximum price, and by forgetting to check you risk a significant fine (depending on the conductor).

3.3 How to deal with failures?

With credit stored on the token, the solution is more fragile and risky for the user. Payment related failures can be dealt with as in solution 1. Other failures are discussed below.

Dispute resolution depends on clear information about what happened about the time a failure occurred. Unfortunately, due to their privacy friendly nature, the protocols retain very little useful information by themselves. Adding timestamps to local logs of each protocol step, by the clearinghouse, the PTO, and the user device will help compare logs in case of disputes (and detect possible fraud). Creating append only logs (using hash chaining techniques) increases their integrity, especially if occasional public commitments to the current state of the log are recorded. A hash of the log on a user device can be submitted when checking in and checking out, and be included in the check in and check out token (that are signed by the PTO). This poses no linkability as the log will be updated with every check in and check out, provided such updates always contain some private information from the user device (e.g. the serial number used in the next credit token).

Check in fails If it is a communication error in the first step, the user can try again. Otherwise, if the credit token fails to verify the user needs to start a dispute resolution (if she believes the credit token should be valid). If the credit token is accepted, but subsequent steps fail, dispute resolution should clear the recorded serial number for the credit token from the clearinghouse database to ensure it is valid the next time the user checks in.

Check out fails If it is a communication error, the user can try again. Otherwise, if the credit token fails to verify the user needs to start a dispute resolution (if she believes the credit token should be valid). If the check in token is not accepted, dispute resolution needs to determine whether the user actually tried to check in earlier, or did not. If the user did not get an error when checking in, for sure the PTO log will contain the serial number of the current credit token.

User does not receive updated credit token This can happen when the user tops up the credit on the device, or when checking out. Retained messages for the credit token issuing protocol may help repair a

Neither is easy to fix: platforms do not have check in poles to quickly jump out and back into the train for example. What is worse: there is no way for users to check whether they checked in or not: PTOs offer no device or service to quickly do so on the platforms or in the trams or busses.

failed run of the credit token (similar to what was discussed for solution 1). If it can be determined that the credit token was really not issued at all, a new credit token can be issued.

Fare dispute After checking out user discovers that the fare paid does not correspond to the fare due for the trip made. The user should submit a piece of the log with all entries involving the check in and corresponding check out for this trip (which should follow each other immediately in the user device log, and are thus linked through the internal hash chain). This is then matched with the corresponding logs of the PTO and clearinghouse. Any discrepancy can be compensated by adding it to the current credit on the device by issuing a new credit token. This can be done even after the user has made other, more recent, trips.

4 Solution 3: Pay as you go, paying later

Given the potential benefits of pay as you go, it is interesting to investigate this idea further. It would especially be nice to allow users to pay for their trips afterwards, instead of having to lock significant funds on the device itself. Also it would be nice to make checking in and out easier or more 'forgiving'. Introducing a pay later option creates a risk for PTOs as users may fail to pay their debts, so mitigation strategies need to be considered.

The basic idea is use the approach from solution 2, allowing negative credit. The main risk is that users use their token up to the maximum debit, and then throw away the old token (or deinstall the app) and then obtain a fresh one with a balance of zero. To counter such sybil-like attacks, obtaining a token should be hard, or should even be tied to your identity. In the latter case the issuer of the tokens can maintain a database of people to which a token has already be issued, and can start asking questions when someone repeatedly requests a new token.

But this is not as straightforward as it seems, because ideally we want to allow arbitrary third parties to provide public transport apps (to increase trust in the privacy properties of the system).

One idea is that any (third party) app must be 'blessed', by the clearinghouse, with an 'admission credential'. In other words, a user can install any app he or she desires, but all protocols outlined above first verify whether the user has a valid admission credential. The user can obtain

this credential, through the app, by registering the app with the clearinghouse[12]. This registration process requires the user to prove his or her identity (for example using a government wide electronic identity scheme like DigiD in the Netherlands).

The credential used for this purpose is special, because it can be blacklisted: the clearinghouse keeps information about all credentials it issued so that when a user wants to obtain a new admission credential (because he or she claims to have lost their phone, reinstalled the app or whatever), then the previous admission credential becomes blacklisted. Information about the blacklisted credential is sent to all PTOs so that when they check whether some user has a valid admission credential (in the first step of each protocol), this will fail for all blacklisted credentials. Note however that this will not deteriorate the privacy protection offered by the protocols, at least not for users without blacklisted credentials: for every credential that is *not* blacklisted, the PTOs have no way to trace or link valid admission credentials that are not yet blacklisted. The exact privacy properties depend on the specific method to blacklist credentials: a naive scheme might allow the clearinghouse to share blacklisting information about *all* users to the PTO to make them all traceable. The most privacy friendly scheme doesn't even allow blacklisted users to be linked or identified [2].

5 Subscriptions, reductions, season tickets etc.

Subscriptions or season tickets, that offer reduced fares, can also be supported as follows.

When using smart cards as tokens this is very easy: the outside of the card contains a picture (binding card holder to the card) and encodes the type of subscription. The type of subscription is also embedded in the card itself as an attribute based credential (that is revealed when buying a ticket or when checking in or checking out) and used to compute the fare, and used by the conductor to verify the fare with the actual trip (when inspecting the ticket in solution 1), or to verify the check in (when inspecting check ins, in solution 2 and 3). Using attribute based credentials to encode subscriptions or entitlements to reduced fares retains the overall privacy properties of the system.

[12] However, there should be a way to tie this credential to the specific device being used, to prevent cloning. See also the discussion in the next section on a way to sidestep this problem.

To support subscriptions or season tickets when smartphones are used as a token, more care needs to be taken. The subscription itself is again easily embedded as an additional credential. But if the public transport app used is not trusted (which is the case if it is running on untrusted hardware, or if it is provided by an arbitrary third party), then there is a risk that people share a single subscription among each other. In other words, there needs to be a way for the conductor to verify whether the subscription credential shown alongside a ticket actually belongs to the user showing it, without making this person linkable. The easiest and most secure as well as privacy friendly approach is probably to stick to a traditional physical pass with a picture of the holder (with certain security features that make counterfeiting hard enough) that travellers need to show in order to prove they have a certain subscription and are eligible for a reduced fare[13].

Alternatively, we can prevent the cloning and sharing of such credentials by making them also one-time use. Instead of encoding a subscription as a persistent attribute based credential, it is encoded as a subscription token using a partially blind signature with a blind serial number and a known description of the type of subscription. Whenever the subscription token is used (when checking in, or buying a ticket), the current one is invalidated using the serial number, and a fresh one is issued. This way, copying a subscription token means a user gives it away: she can no longer use it herself. It does still allow a few users to share a credential 'serially': one person can use it at a time, and after use give it to someone else to use.

[13] The second *meta conclusion* of this work is that there is a need to make apps (or data in apps) *uncloneable*, so that they can be used in similar contexts and with similar properties as smart cards. Moreover, there should be a secure way to establish that the person holding the phone and/or using the app is indeed the owner of the phone (and not someone that uses the phone with permission of the real owner). These properties are also mandatory to increase the security of attribute credentials, in particular to prevent the attributes in them being pooled or shared. This seems challenging if at the same time we want the apps to be open source… One idea is to use either the SIM card present in most smartphones (but this requires cooperation with the mobile network operators), or to use the secure modules present in most smartphones.

6 Conclusions

In this (preliminary) note we explored some options to implement privacy friendly ticketing for public transport. We show that this certainly possible, with certain constraints (or issues that deserve further study).

Starting point is the observation that from a privacy perspective it is better to collect personal data locally on the user device, instead of centrally on the servers of the service providers.

Three different approaches (buying tickets beforehand, pay as you go, and pay after you went) have been studied. We show that these can be implemented with reasonably good privacy properties, under reasonably practical assumptions. In particular we show that an untrusted smartphone can be used as the 'token' to carry tickets or travel credit. This allows third parties to provide the apps for that purpose, which should increase the (perceived) trust of the overall system. At the very least it offers users a real choice.

For the third, pay after you went, solution (where users pay afterwards after making one or more trips), public transport operators bear a risk that requires further study.

Also, there are rather strict requirements on the maximum checking in and checking out time (in the order of 200-300 milliseconds); actual implementations of the protocols proposed are necessary to verify whether these requirements can be met.

Finally, further analysis of the possible ways in which these protocols can fail and how that affects the user experience (and PTO risks) is also required.

7 Acknowledgements

I'd like to thank Hanna Schraffenberger and Sietse Ringers for useful comments and suggestions.

References

[1] Masayuki Abe and Tatsuaki Okamoto. "Provably Secure Partially Blind Signatures". In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. 2000, pp. 271–286.

[2] Jan Camenisch and Anna Lysyanskaya. “Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials”. In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. 2002, pp. 61–76.

[3] David Chaum. “Blind Signatures for Untraceable Payments”. In: *Advances in Cryptology: Proceedings of CRYPTO ’82, Santa Barbara, California, USA, August 23-25, 1982.* 1982, pp. 199–203.

[4] Nationaal Openbaar Vervoer Beraad (NOVB). “Visie OV Betalen. Een verkenning naar de OV betaaltechnieken van de toekomst”. http://cooperatieovbedrijven.nl/wp-content/uploads/2019/05/Visie-OV-Betalen.pdf. Dec. 2014.

[5] Tatsuaki Okamoto. “Efficient Blind and Partially Blind Signatures Without Random Oracles”. In: *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. 2006, pp. 80–99.

[6] Rafael Pass. “On Deniability in the Common Reference String and Random Oracle Model”. In: *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. 2003, pp. 316–337.

[7] Stuart G. Stubblebine, Paul F. Syverson, and David M. Goldschlag. “Unlinkable serial transactions: protocols and applications”. In: *ACM Trans. Inf. Syst. Secur.* 2.4 (1999), pp. 354–389.

[8] IBM Research Zürich Team. *Specification of the Identity Mixer Cryptographic Library*. report. Version 2.3.4. Zürich: IBM Research.