Secure Anonymous Signature-Based Transactions

Els Van Herreweghen

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland
evh@zurich.ibm.com

Abstract. Electronic commerce protocols often require users to reveal their identities and other information not necessary for reasons of security. Some applications such as contract signing are often argued to require a signer's authenticated identity; but this authentication may give the recipient a false feeling of security if certificate registration procedures do not guarantee a mapping to a liable person, or correctness of certificate data. In this paper, we propose a separation of identity from liability. Liability-aware certificates allow certificate issuers to make explicit which liabilities it takes with respect to the transaction, the certificate data or the signer's identity. We illustrate their use in the design of a pseudonym service providing pseudonym certificates for secure anonymous transactions.

1 Introduction

Many electronic commerce services and protocols are not designed with the goal of protecting the privacy or anonymity of end users. In fact, they often require the user to give a lot of information not strictly necessary for reasons of security. Such information can be present in the certificates certifying the user's public key.

As the collection and exploitation of information becomes more of a concern, users are less willing to give out information. It is therefore desirable to re-examine the need for giving out certain information items as part of business processes. Necessary is only the information from which the recipient of a digital signature (representing a payment, an auction bid, a contract) derives trust that the payment is valid, that the bidder will pay upon winning the bid, that the signer is liable to execute the contract. Most often, this trust is not based on the signer's identity. Rather, it is based on the identity and trustworthiness of the authority certifying the signer's key, thereby implicitly or explicitly 'vouching' for transactions made with that key.

Also, the liability of certification authorities with respect to certificate registration procedures is often unclear. When a party relying on a signature requires that it be made by a certified 'real' identity, there may be an assumption but no guarantee that a legal person can be held liable for this signature. Rather than receiving a signature by an authenticated 'real' identity certified under uncertain liability rules, it would be more useful for the relying party to have the certifier's guarantee that a legal person can be held liable for the signature.

In this paper, we demonstrate a separation of identity from liability and certification. We transform a number of signature-based protocols into pseudonymized versions. We show how to provide maximal security for the relying party by including issuers' liabilities into signers' certificates. We distinguish between liability for data in the certificate, liability for transactions made with the certificate, and liability to reveal a signer's real identity under specified conditions.

The outline of the paper is as follows. In Section 2, we introduce the concept by describing a pseudonym server transforming certificates in a generic account-based payment system into single-use pseudonym certificates. We show that it is possible for the payment recipient to have a guarantee of payment, without affecting the protocol, and without introducing any liability for the pseudonym server. In Section 3, we discuss the value of certificates and signatures for more general applications, and introduce a liability-aware certificate format. In Section 4, this certificate format is used in the design of a generic pseudonym server, issuing pseudonym certificates for a potentially large set of applications. Section 5 discusses related work and suggestions for future research; Section 6 concludes the paper.

2 Pseudonymizing a Generic Payment System

The generic payment protocol in Figure 1 follows the model of the iKP [1,2] and SET [3] protocols. Its exact format, however, is derived from the requirements for disputable payment protocols in [4]. After presenting the protocol, we analyze its implicit guarantees and liabilities. We then introduce a pseudonym server (PS) into this system and design a pseudonymized version of the payment protocol. The goal is to preserve existing payment guarantees for the relying party while minimizing the pseudonym server's liability.

2.1 The Generic Payment Protocol

The participants in the protocol are C (Customer), M (Merchant) and A (Acquirer). C's certificate $\mathrm{CERT_C}$ is issued by I (Issuer) and specifies I's liability for payments made using $\mathrm{CERT_C}$; I's certificate $\mathrm{CERT_I}$ is assumed to be issued by a root certification authority.

We use the following notation:

$\mathrm{SK_X}$, $\mathrm{PK_X}$: X's secret signature key and public key in a digital signature scheme.
$\mathrm{S_X(M)}$: Signature with $\mathrm{SK_X}$ over message M – does not include M.
$\mathrm{E_Y(M)}$: Encryption of M under Y's public encryption key.
$\mathrm{CERT_X}$: X's public-key certificate (certifying $\mathrm{PK_X}$).
role: Role in the payment system ('customer', 'merchant', 'issuer', 'acquirer').

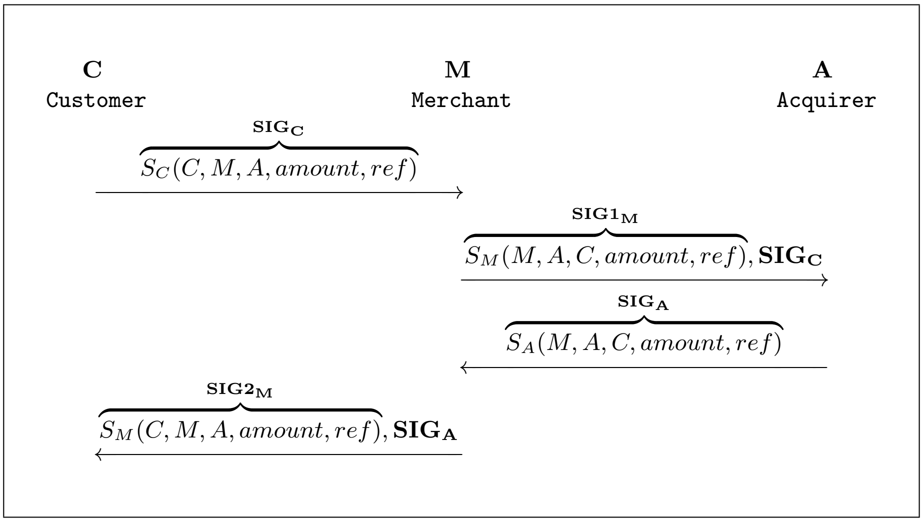
C
Customer
M
Merchant
A
Acquirer
$\overbrace{S_C(C, M, A, amount, ref)}^{\mathbf{SIG_C}}$
$\overbrace{S_M(M, A, C, amount, ref)}^{\mathbf{SIG1_M}}, \mathbf{SIG_C}$
$\overbrace{S_A(M, A, C, amount, ref)}^{\mathbf{SIG_A}}$
$\overbrace{S_M(C, M, A, amount, ref)}^{\mathbf{SIG2_M}}, \mathbf{SIG_A}$

Fig. 1. Generic payment protocol

With $\mathrm{SIG_C}$, C authorizes a payment to M, who then adds its own signature $\mathrm{SIG1_M}$ to form an authorization request. $\mathrm{SIG_A}$ constitutes A's authorization response. M creates a payment receipt $\mathrm{SIG2_M}$ and sends this together with $\mathrm{SIG_A}$ to C.

The parameter ref contains at least a transaction identifier trx_id and a transaction time, possibly also a goods description or a hash thereof. We only represent the actual signatures; which of the components of a signature are actually sent as part of the protocol messages, or as part of a previous negotiation protocol, is not relevant to this discussion: it is assumed that the verifier of a signature possesses the data constituting the signed message (including the necessary certificates to verify them).

M may verify $\mathrm{SIG_C}$ to validate the correctness of transaction data. M's payment guarantee, however, is derived from A's authorization $\mathrm{SIG_A}$. A's authorization, in turn, is based on the validity of $\mathrm{SIG_C}$, the contents of $\mathrm{CERT_C}$ and $\mathrm{CERT_I}$, and the terms of the contract between A and I. The combination of $\mathrm{CERT_C}$, $\mathrm{CERT_I}$ and this contract provide A with a guarantee such as "I will honor validly constructed payments with $\mathrm{PK_C}$ up to an amount of \$1000 subject to the condition that $\mathrm{CERT_C}$ nor $\mathrm{CERT_I}$ were on a revocation list accessible by A at the time A authorized the transaction". As I may not be able to verify exactly when A authorized the payment, I may allow a maximum processing delay : "...were on a revocation list 12 hours before A presents the transaction to I for clearance".

Liabilities or revocation conditions may not actually be part of existing certificates; rather, they are part of contracts, implicit agreements or implicit procedures and are thus implicitly present in a certificate. We can then represent a certificate $\mathrm{CERT_X}$, issued by Y [1]:

$$\mathrm{CERT_X = S_Y(role,\ X,\ PK_X,\ Y,\ L = \{amount,\ condition\})}$$

where L stands for Y's liability for transactions made with $\mathrm{PK_X}$. L consists of an amount and possibly the conditions related to revocation.

In the example above, assuming that I is directly certified by a root authority:

$$\mathrm{CERT_C = S_I(customer,\ C,\ PK_C,\ I,\ L = \{amount{=}\$1000,\ maxdelay = 12\ hrs\})}$$
$$\mathrm{CERT_I = S_{root}(issuer,\ I,\ PK_I,\ root,\ L = \{amount{=}\$10{,}000,\ maxdelay = 12\ hrs\})}$$

2.2 Requirements for a Secure Pseudonymized Version

We now develop a pseudonymized version of the above generic payment protocol. C, possessing a long-term payment certificate $\mathrm{CERT_C}$, can obtain one-time pseudonym certificates $\mathrm{CERT_P}$ from a pseudonym server PS, allowing C to make payments under the pseudonym P and remain anonymous towards M and A. (We assume that we cannot change the existing payment infrastructure and thus that the interface between C and I, such as registration, account definition and payment processing, is fixed. In this infrastructure, C has a non-anonymous account with I, and I expects a valid payment order to be linked to that account. Therefore, we do not consider anonymity of C towards I. However, the constructions introduced to anonymize C towards M and A can also be applied to anonymize C towards I if we relax the previous assumption and allow a change to the interface between C and I.) PS, of course, needs to be a recognized issuer in the specific payment system in order for $\mathrm{CERT_P}$ to be considered a valid certificate by the relying party.

We first list the different criteria against which we will measure the pseudonymized version.

Guarantees towards relying party (A). A valid signature $\mathrm{SIG_P}$ over a transaction has the same guarantee of payment as a valid signature $\mathrm{SIG_C}$ over the same transaction.

For general signature-based protocols (as discussed in Sections 3 and 4), a relying party may want the pseudonym certificate to express the conditions (e.g., fraudulent behavior) under which it can obtain a real identity of the signer. In the specific case of a payment protocol, however, we assume that only the guarantee of payment is relevant to the relying party. It should therefore be unnecessary to ever reveal the real identity of even a dishonest or fraudulent C to A (or M).

[1] The notation for liability is intuitive and does not define a specific format or language. The certificate format is merely a symbolic representation; additional information such as expiration time, attributes etc. are represented only when relevant to the discussion.

No extra risk for I. I wants proof of C's payment authorization even if the payment is made under pseudonym P: if I does not trust PS, I has no reason to debit C's account for pseudonymous payments under a pseudonym P which allegedly represents C. Even if I trusts PS (e.g., I operates PS as a service), I may need to prove C's authorization to a third party, e.g. in order to protect itself against repudiation by C.

Minimal trust by C in PS. PS should protect C's privacy and not give away information about C, or reveal the linking between C and the pseudonyms it uses. PS may publish privacy policies stating to which end, under which conditions and to whom PS may reveal which information about C. As mentioned in the first requirement, PS should never have to reveal any information about C to M or A. (Of course, if PS ever has a dispute with C, PS may need to reveal C's identity to a third party.) It is difficult to enforce and verify the PS's compliance with its privacy policies; C has to trust PS to adhere to them. This requirement holds for the various pseudonym server scenarios in this paper, and we will not repeat it when discussing their security.

Another of C's requirements is for PS not to be able to frame C. This requirement seems easy to fulfill if C is allowed to issue its own pseudonym private keys. We will see, however, that PS may have an interest in issuing the pseudonym private keys if it wants to limit its own risk.

Minimal risk for PS. PS is liable for transactions made with certificates it issued. PS can rigorously limit that liability, e.g., by issuing single-use $\mathrm{CERT_P}$ only after receiving a valid $\mathrm{SIG_C}$ from C. Then, when PS is held liable for a payment ($\mathrm{CERT_P}$, $\mathrm{SIG_P}$), it can show (to I or a to a third party) a payment ($\mathrm{CERT_C}$, $\mathrm{SIG_C}$) for which I is liable. Still, revocation issues need to be taken into account: for PS to assume no liability at all, the possibility should be excluded that $\mathrm{CERT_P}$ can be valid after $\mathrm{CERT_C}$ has been revoked.

In the following sections, we investigate different possibilities for pseudonymizing the previous protocol, with different impact on liability of PS, on efficiency, and on PS's infrastructure requirements for revocation.

2.3 Design for Maximum Security: PS Online, $\mathrm{CERT_P}$ Linked to Transaction

In this first design, PS issues a different pseudonym key pair ($\mathrm{PK_P}$, $\mathrm{SK_P}$), pseudonym certificate ($\mathrm{CERT_P}$) and signature $\mathrm{SIG_P}$ for C for each transaction. PS is thus always on-line and participates in the transaction. The pseudonym certificate is linked to the specific transaction; this guarantees that it can be used only once.

Description of the protocol. The P-M-A transaction in the pseudonymized protocol is the same as the C-M-A transaction in the original protocol, with

C's identity, signature and certificate replaced by P's identity, signature and certificate. M and A see P as customer; and PS as the issuer of P's certificate:

$\mathrm{SIG_P} = \mathrm{S_P}(\mathrm{P,M,A,time,amount,ref})$
$\mathrm{SIG1_M} = \mathrm{S_M}(\mathrm{M,A,P,time,amount,ref})$
$\mathrm{SIG_A} = \mathrm{S_A}(\mathrm{M,A,P,time,amount,ref})$
$\mathrm{SIG2_M} = \mathrm{S_M}(\mathrm{P,M,A, time,amount,ref})$

PS issues $\mathrm{SIG_P}$ only after having received and validated a transaction signature $\mathrm{SIG_C}$ from C. $\mathrm{SIG_C}$ is PS' proof of C's commitment to the transaction. PS includes $\mathrm{SIG_C}$ in $\mathrm{CERT_P}$, allowing I to check $\mathrm{SIG_C}$: this protects PS against repudiation by C and protects C against framing by PS.

New certificates introduced in the system are then:

$\mathrm{CERT_{PS}}$ is a root-issued issuer certificate:
$\mathrm{CERT_{PS}} = \mathrm{S_{root}}(\mathrm{issuer, PS, PK_{PS}, root, L_{root}})$.
$\mathrm{CERT_P} = \mathrm{S_{PS}}(\mathrm{customer, P, PK_P, PS, L_{PS}, issuer\_data})$

with issuer_data = $\mathrm{E_I}(\mathrm{SIG_C, C})$, the encryption of C's identity and transaction signature. It can only be decrypted by I and provides I with proof of C's authorization.

The resulting protocol is depicted in Figure 2. To highlight the difference with the original version, we also show the transport of certificates belonging to C (and I) and P (and PS).

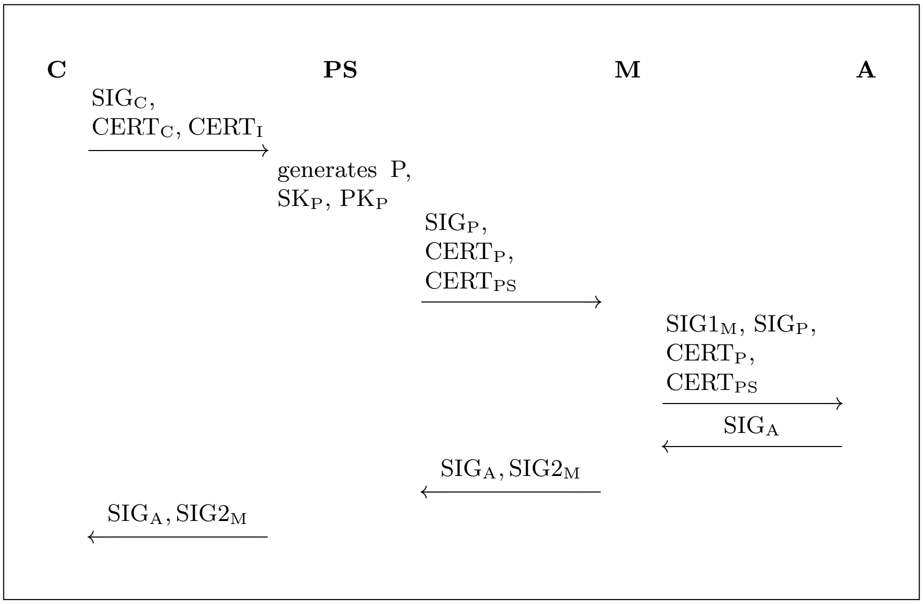

Fig. 2. Pseudonymized payment protocol: on-line PS, one-time pseudonym certificate

The settlement between A and I (not shown in the figure) may be done at a later point in time. Depending on A's 'awareness' of pseudoymized certificates, A may initiate a settlement with PS by sending {$\mathrm{SIG_P}$, $\mathrm{CERT_P}$} to PS and PS sending issuer_data to I. Alternatively, A directly sends issuer_data to I.

To hide the linking between C and P also towards outsiders, the channel between C and PS additionally needs to be encrypted. Also, traffic anonymizers or MIXes [5,6,7] used on the C-PS and C-M channels can help to unlink the pseudonym request from the pseudonym use, and to hide the linking of C or P to a network address. These extensions apply to the different pseudonym server designs in this paper but are not discussed here.

The 'receipts' $\mathrm{SIG2_M}$ and $\mathrm{SIG_A}$ don't include C's identity; C may thus not be able to use them as transaction receipts. One could allow C to prove the linking between C and P by including such a proof (e.g., a hash H(C,P,ref)) in $\mathrm{CERT_P}$ and by PS returning $\mathrm{CERT_P}$ to C together with $\mathrm{SIG_A}$ and $\mathrm{SIG2_M}$. Still, this solution requires C to reveal its real identity when proving its participation in the transaction. This can only be avoided by changing the way receipts are used. E.g., C could prove ownership of a receipt $\mathrm{SIG2_M}$ or $\mathrm{SIG_A}$, not by proving a linking between C and P but by dynamically proving knowledge of a secret associated with P while showing the receipt. Such a secret could be $\mathrm{SK_P}$ or another secret key associated with P, and would be securely communicated by PS to P. This alternative is a topic for future research.

Security analysis. We now analyze the above protocol in terms of the requirements in Section 2.2.

Guarantees towards relying party (A). We define a valid ($\mathrm{SIG_P}$, $\mathrm{CERT_P}$, $\mathrm{CERT_{PS}}$) to constitute the same payment guarantee as in the original system, i.e., PS is liable to honor a valid $\mathrm{SIG_P}$ under the condition of $\mathrm{CERT_P}$ and $\mathrm{CERT_{PS}}$ not being revoked.

No extra risk for I. I only clears payments based on a valid $\mathrm{SIG_C}$, proof of C's transaction commitment.

Minimal trust by C in PS. PS cannot frame C, as I only accepts payments containing a valid $\mathrm{SIG_C}$.

Minimal risk for PS. PS does not take any risk at all as long as it only issues $\mathrm{CERT_P}$ for a valid $\mathrm{SIG_C}$ and checks that $\mathrm{CERT_C}$ is not on any revocation list. This ensures that any valid $\mathrm{SIG_P}$ is based on a valid $\mathrm{SIG_C}$, i.e., any payment which is acceptable to A will be honored by I, and thus PS can transfer any liability to I. PS only increases its risk (up to the amount in $\mathrm{CERT_P}$) by not acting honestly or correctly.

The absence of risk for PS strongly depends on PS issuing the pseudonym key pair. If C were allowed to issue his own ($\mathrm{SK_P}$, $\mathrm{PK_P}$), it would be possible for C to generate a $\mathrm{SIG_P}$ inconsistent with $\mathrm{SIG_C}$, causing liabilities for PS not covered by I.

Discussion. The model of PS issuing pseudonym keys and certificates for every new transaction provides maximum security to all the parties, and a total absence of risk for the PS. For reasons of efficiency, however, it may be desirable for C to obtain pseudonym certificates ahead of time, such as to avoid contacting PS for every transaction. The next section describes such an alternative solution and its security features.

2.4 Alternative Design: Offline PS

Allowing C to obtain certificates ahead of time has two direct consequences. First, the pseudonym certificate can no longer be linked to a specific transaction, and thus has to be valid for a certain amount of time. Second, as C makes the actual pseudonym payment without PS's involvement, C has to issue the pseudonym key pair. This introduces major risks:

1. PS cannot enforce a linking between $\mathrm{SIG_C}$ and $\mathrm{SIG_P}$. Or, PS cannot enforce that P's payment valid for A contains C's payment acceptable to I.
2. After issuing $\mathrm{CERT_P}$, $\mathrm{CERT_C}$ can be revoked, which leaves PS with the liability for payments with $\mathrm{CERT_P}$ until $\mathrm{CERT_P}$ is revoked or no longer valid.

The second problem can be addressed in either of the following ways:

- PS issues very short-lived pseudonym certificates (e.g., $\mathrm{CERT_P}$ lifetime is smaller than the revocation delay of $\mathrm{CERT_C}$). Then, even if $\mathrm{CERT_C}$ is revoked, any outstanding $\mathrm{CERT_P}$ will be invalid by the time I refuses payments with $\mathrm{CERT_C}$. This solution, however, defeats the purpose of C obtaining certificates some time before the actual transaction.
- PS revokes pseudonym certificates by posting Certificate Revocation Lists (CRLs) to the appropriate CRL directories. It should then frequently verify revocation status of parent certificates $\mathrm{CERT_C}$ of outstanding pseudonym certificates $\mathrm{CERT_P}$, and revoke pseudonym certificates as needed.

In order to address the first problem, PS has to take the role of an insurer, resort to risk management strategies, and charge for the service accordingly. Alternatively, PS may require a deposit or guarantee from C equal to PS's liability for the pseudonym certificate. Neither solution, however, gives I or a third party a proof of transaction by C. The protocol in Figure 3 constructs such a proof by C committing to the pseudonym (keys) as part of the pseudonym retrieval procedure; this commitment $\mathrm{S_C(CERT_C, PK_P, \ldots)}$ may be encrypted for I and included in $\mathrm{CERT_P}$, in a way similar to $\mathrm{SIG_C}$ in the protocol in Figure 2. Such a scenario changes the dispute handling rules of the protocol: with the combination of $\mathrm{SIG_P}$ and the commitment, I as well as an external verifier should conclude that C authorized the payment. It may also solve the first problem: if PS includes the same liability in $\mathrm{CERT_P}$ as in $\mathrm{CERT_{PS}}$, PS again can transfer liability to I for every payment valid to A.

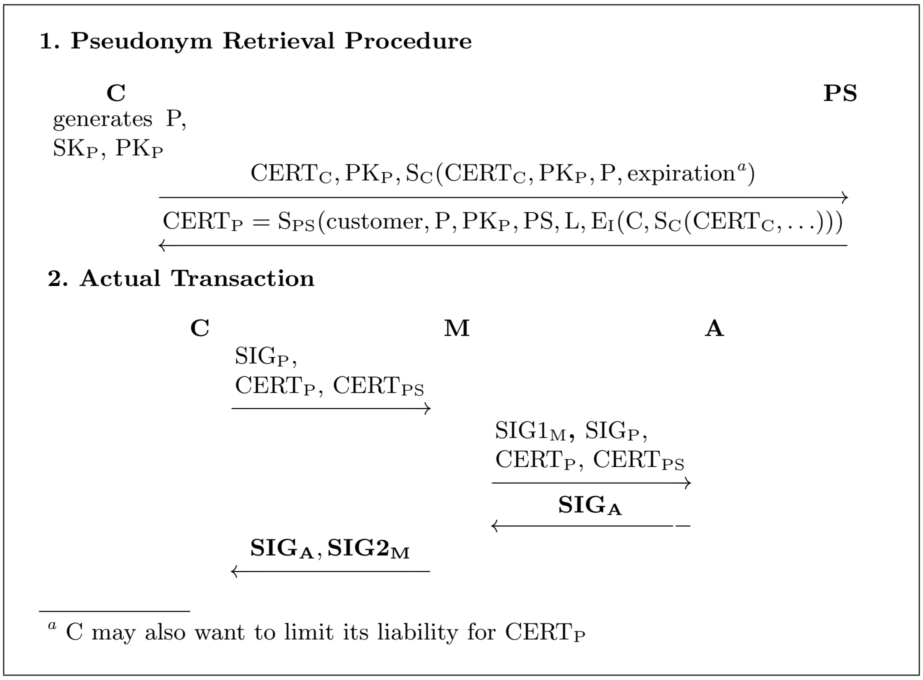
1. Pseudonym Retrieval Procedure
C
PS
generates P, SK_P, PK_P
$CERT_C, PK_P, S_C(CERT_C, PK_P, P, expiration^a)$
$CERT_P = S_{PS}(customer, P, PK_P, PS, L, E_I(C, S_C(CERT_C, \ldots)))$
2. Actual Transaction
C
M
A
SIG_P, $CERT_P$, $CERT_{PS}$
$SIG1_M$, SIG_P, $CERT_P$, $CERT_{PS}$
SIG_A
$SIG_A, SIG2_M$
[a] C may also want to limit its liability for $CERT_P$

Fig. 3. Pseudonymized payment protocol: off-line PS

Security Features

Guarantees towards relying party (A). We define, as in the previous solution, a valid (SIG_P, $CERT_P$, $CERT_{PS}$) to represent a payment guarantee for A, on condition that $CERT_P$ or $CERT_{PS}$ are not revoked.

No extra risk for I. From $CERT_P$ and the encrypted commitment, I can derive C's identity and commitment to P and PKp; from SIG_P, P's authorization of the transaction. Thus I takes no extra risk on condition that a potential external verifier applies the same verification rules.

Minimal trust by C in PS. PS cannot frame C, as C generates the pseudonym keys.

Minimal risk for PS. If PS takes care of the revocation problem (immediately revoking $CERT_P$ in case of a revoked $CERT_C$) and makes sure not to take more liability in $CERT_P$ than I takes in $CERT_C$, PS again limits its liability.

2.5 Discussion

The designs in Sections 2.3 and 2.4 both provide equal guarantees towards the relying party but show some tradeoffs between PS's liability, and changes of infrastructure and efficiency. An off-line PS serving pseudonym certificates ahead of time may be a more efficient solution. In this case, PS has to limit its liability by taking care of timely revocation of pseudonym certificates, and by requiring that the user commit to the pseudonym key. This commitment has to be recognized by an external verifier in order for I to prove the user's authorization through P.

An on-line PS issuing transaction-linked certificates is a less efficient solution but allows PS to rigorously minimize its liability without requiring any changes in verification and dispute handling infrastructure, and without requiring PS to handle revocation issues.

3 Generalized Signatures and Liabilities

The pseudonym server designs in the previous section are specific for the payment protocol and assume that the user already has a long-term certificate for use in this protocol.

When dealing with more general signatures, such as digitally signed contracts, it may no longer be possible to attach a fixed value to the transaction, as we did in the case of payments. The party relying on a signed contract may also attach a value to the guarantee that the signer is a liable person whose identity will be revealed under certain conditions, or to the correctness of certain attributes in the signer's certificate.

A generic digital signature (with corresponding certificate) can be represented as:

$$\mathbf{CERT_S} = \mathrm{S_I}(\mathrm{Pid, role, S, PK_S, I, attrs}),\ \mathbf{SIG_S} = \mathrm{S_S}(\mathrm{trx\_data, trx\_id})$$

with

Pid, role: the protocol identifier and S's role;
S: the signer (of the contract, auction bid, payment, ...);
I: the certificate issuer certifying $\mathrm{PK_S}$;
trx_data: describing the type and contents of the transaction;
trx_id: a unique transaction identifier (similar to the ref parameter).

$\mathrm{SIG_S}$ expresses S's commitment either to the correctness of trx_data, or to the execution of some action described in trx_data, such as payment or contractual obligation. A verifier V can hold S (or, I through $\mathrm{CERT_S}$) liable for this commitment. In order to determine the potential value of such a liability, we describe the different damages to V that can be incurred:

– V may suffer loss if attrs in $\mathrm{CERT_S}$ are not correct; e.g., attrs specify a quality label of S as a registered construction company and SIGs represents a bid for constructing a building.

– V may suffer loss if data in trx_data is incorrect; this may be relevant if S's commitment is a commitment to data rather than to an action or contract.
– If trx_data represents an action or contractual obligation, V suffers loss if the promise (contract) expressed in trx_data is not kept (executed). In the special case of the contract being a payment, it can be the value of the payment.

In order to express I's or S's liability for data (attrs or trx_data), it should be possible to attach a value to their correctness. Also, their correctness should be publicly verifiable, or, it should be specified which third party or arbiter can judge this.

3.1 Liability-Aware Certificates

The possible losses described above can be covered by liabilities included into certificates:

$$\mathrm{CERT_S} = \mathrm{S_I}(\mathrm{Pid, role, S, PK_S, I, attrs, L})$$

L, the issuer's liability, consists of a set (0 or more) of individual components, each of which have one of the following types: Ld (related to data liability), Lt (related to transaction liability) and Li (related to identity liability). Other than a type, each component has an amount, a condition for liability, and an indication of which party can evaluate the condition.

Some examples of individual liability components are:

– a liability {type = Ld, amount = \$1000, condition = attrs, verifier = Arbiter} indicates that the certificate issuer takes a liability of \$1000 for any certificate attributes which are found to be incorrect by Arbiter.
– a liability {type = Ld, amount = \$1000, condition = trx_data, verifier = Arbiter} indicates that the certificate issuer takes a liability of \$1000 for any trx_data signed by the certificate holder which are found to be incorrect by Arbiter.
– a liability {type = Lt, amount = \$1000, condition = any, verifier = Arbiter} indicates that the certificate issuer takes a liability of \$1000 if Arbiter evaluates the transaction (e.g., the contract execution) as failed.
– a liability {type = Li, amount = \$10000, condition = trx_data, verifier = Arbiter} indicates that the certificate issuer is liable for revealing the real identity of a signer (who is a liable person) if Arbiter evaluates trx_data to be false or not executed. Failure to reliably reveal a real identity makes the issuer liable to compensate \$10000.

Following are some additional examples of the use of liabilities in specific applications:

– A liability in a user certificate for a SET-like payment system could look as follows:

L = {{type=Lt, amount=$1000, condition=(revocation_delay=12hrs), verifier=Arbiter}}.
Here, the only liability taken by the issuer is to honor payments up to $1000 as long as the revocation condition is fulfilled. The issuer does not take any liability for attributes or for revealing a signer's identity.

– A liability in a certificate of a construction company used for contract signing could be:
  L = {{type=Ld, amount=$1000, condition=attrs, verifier=Arbiter}, {type=Li, amount=$10000, condition=trx_data, verifier=Arbiter}}.
  The issuer takes no liability for the transaction (or its execution); it takes a liability of $1000 for any false attributes, and guarantees, with a penalty of $10000, to reveal the real name of a signer if the contract is not executed as asserted by the verifier.

The above examples are meant to demonstrate the concept; in a detailed design, the expression of individual liabilities and conditions may require more complex statements.

In the following section, we now describe the design of the pseudonym server. We use the liability-aware notation for long-term and pseudonym certificates.

4 A Generic Pseudonym Server

We would like a generic pseudonym server to fulfill following requirements:

1. The pseudonym server issues certificates that can be used in a set of different protocols, such as contract signing, auction, payment.
2. The pseudonym server can serve certificates in a protocol-independent manner, i.e., does not play an active role in composing protocol-specific signatures.
3. The pseudonym certificates are not necessarily linked to a specific transaction, allowing for pseudonym certificates to be acquired ahead of time.
4. The one-time user of a pseudonymous service or protocol need not have a long-term certificate for that protocol, i.e., a user registered with the pseudonym server may request one-time pseudonym certificates for a number of protocols. E.g., the PS could be a service provided by a user's Internet Service Provider (ISP) and the user can obtain a one-time SET certificate for a specific payment.

A user S of such a pseudonym service has a long-term certificate $\mathrm{CERT_S}$ recognized by or issued by PS, and can obtain pseudonym certificates from PS for different protocols and transactions. Figure 4 shows such a pseudonymized transaction. The CERT_REQ - CERT_RES exchange constitutes the core PS functionality. The other flows illustrate the possible use of pseudonymized certificates within a specific transaction: in an optional CERT_PROP flow, the verifier V sends to S the proposed certificate format; the last flow containing $\mathrm{SIG_P}$ is S's actual transaction commitment using pseudonym P.

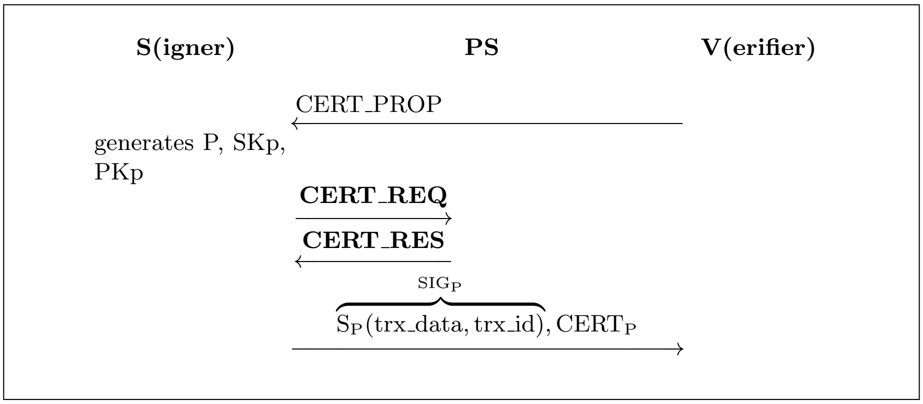

Fig. 4. Generic pseudonym server

4.1 CERT_REQ, CERT_RES

We now describe the message and certificate formats in more detail; the design choices are then clarified when analyzing the security of the pseudonymous transaction.

$\mathrm{CERT_S = S_I(pid_S, role_S, S, PK_S, I, attrs_S, L_{I-S})}$

with $\mathrm{L_{I-S}}$ the set of liabilities I takes for data in $\mathrm{CERT_S}$ and transactions with $\mathrm{CERT_S}$.

$\mathrm{CERT_P = S_{PS}(pid_P, role_P, P, PK_P, PS, attrs_P, L_{PS-P})}$

with $\mathrm{L_{PS-P}}$ the set of liabilities PS takes for data and transactions with $\mathrm{CERT_P}$.

$\mathrm{CERT_P}$ may be linked to a specific transaction by specifying a transaction identifier as part of the various liability conditions. This limits the risk taken by PS, and may make it unnecessary for PS to have to deal with revoking $\mathrm{CERT_P}$. Note that PS can be the same entity as I, in which case S simply has a long-term account with PS. This can be the case if the PS service is offered by a ISP.

$\mathrm{CERT\_REQ = S_C(CERT\_REQ, pid_P, role_P, P, PK_P, PS, attrs_P, L_{PS-P}), CERT_S}$
$\mathrm{CERT\_RES = CERT_P}$

With CERT_REQ, S communicates the desired certificate contents to PS, and at the same time commits to signatures made with SKp in the context of the requested $\mathrm{CERT_P}$. It can thus be seen as a secure registration procedure where S takes liability for $\mathrm{PK_P}$ and the $\mathrm{CERT_P}$ not yet received, in a way provable by PS.

PS's decision to issue $\mathrm{CERT_P}$ depends on its evaluation of risk. This depends on the liabilities in the requested $\mathrm{CERT_P}$, whether it is valid for only one or more transactions, the contract terms between S and PS, whether or not S has an account with PS, etc. The processing of CERT_REQ, therefore, may involve additional communication between S and PS, such as PS asking for some guarantee or deposit.

4.2 The Liabilities in CERT_P

In some cases, the expected liabilities in CERT_P can be derived by V and/or S from transaction features, such as the amount of a payment or auction bid. More complex liabilities may be determined by V on a per-transaction basis. V may have to explicitly communicate expected liabilities to S, as shown with the CERT_PROP flow in Figure 4.

Such a message could have a similar format as a CERT_REQ and can be signed by V:

$$\text{S}_\text{V}(\text{CERT\_PROP},\ldots,\text{L})$$

in which case it may a binding commitment, part of the contract between S (P) and V.

Security Features. We now analyze the CERT_REQ - CERT_RES flows with respect to the security features listed in Section 2.2, omitting the issuer-specific requirement which was specific for the payment system.

Guarantees towards relying party V. The relying party V receiving a signature SIG_S in a transaction has a guarantee of I's liabilities in CERT_P, thus of receiving liability amounts or the real identity of the signer.

Minimal trust of C in PS. As S generates pseudonym keys and commits to them using its private signature key SK_S, PS is not able to frame S.

PS absorbs minimal risk. In case of a dispute, PS can prove S's commitment to a pseudonym key, and thus S's involvement in a transaction with that key. The remaining risk depends on the possibilities for PS to 'recover' liabilities from I or S. This depends on contracts between S and PS (or between S and I).

4.3 Example: Auction

We illustrate the above principles with one more example. In this example, an ISP plays the role of PS for its registered users, billing them for this service through the existing billing relationship. Registered users have a limited liability credit with the PS. A user S, registered with the ISP, wants a pseudonym certificate to bid on a last minute vacation offer with an auction service. The auction service accepts anonymous or pseudonymous bids under the condition that payment is guaranteed, the bidder's age is above 18, and the winning bidder's identity will be revealed to the auction service.

S has a long-term certificate CERT_S and private key SK_S with which to authenticate to the ISP/PS. This long-term certificate may contain values of (or references to) maximum liabilities PS is willing to take for this user, attributes of the user (such as age) which PS has verified upon registration and is willing to vouch in a pseudonym certificate.

The pseudonym certificate (and, similarly, CERT_REQ) could look as follows:

$\text{CERT}_\text{P} = \text{S}_\text{PS}$(CERT_REQ, e-auction, bidder, P, PK_P, PS, attrs={{age>18}}, L={{Ld, \$100, none, Arbiter},{Lt, \$300, none, Arbiter}, {Li, \$300, condition(auction_id), Arbiter}}).

This expresses that PS takes a liability of \$100 for the correctness of age, gives a payment guarantee of \$300 for the transaction, and guarantees to reveal S's identity on condition of winning the auction with identifier auction_id. Assuming that PS can prove S's age to be over 18 to Arbiter, and PS has proof of S's real identity, the risk taken by PS is limited to the transaction amount (\$100), which may be below S's credit limit, in which case PS doesn't need additional deposits or guarantees from S.

5 Related and Future Work

In this paper, we focused on existing systems based on X.509 [8] or similar certificates, on introducing explicit liability statements into these certificates, and on optimal control of the pseudonym server over its liabilities. Though the pseudonym server cannot frame users, it still has to be trusted with a large amount of information about users, their transactions, and the linking of their pseudonyms. Pseudonym systems based on blind signatures [9,10]) can provide stronger anonymity guarantees and require less trust in the pseudonym server than the designs presented in this paper. They can provide protection against specific forms of user misbehavior (such as double-spending a coin [11]) through cryptographic constructs that cause the identity of a misbehaving user to be revealed. It is interesting to investigate to which extent such pseudonym systems can support the functionality we provided in this paper. The potential complexity of liability conditions (such as the ones for revealing identities), however, seems to suggest the existence of a trusted party evaluating them, which does not fit with blind signature schemes. Also, blind signatures may make it much harder for an issuer or pseudonym server to control the usage of a pseudonym certificate and thus its liability.

This paper has introduced a concept rather than a full design, and leaves several topics open for investigation. The liability types may need to be refined as more complex electronic negotiations and transactions are defined. Expressing liability contents, such as conditions and verifiers, was done using an intuitive notation; a language and notation need to be developed.

The choice to use pseudonyms as identifiers in certificates was made for reasons of compatibility with the existing certificate formats. It would be possible to remove also those pseudonyms from certificates and use strictly anonymous certificates.

It should also further be investigated how to use traffic anonymizers (such as [5]) or MIXes [6,7] to provide unlinkability between the request for and the actual use of a pseudonym.

6 Conclusion

In this paper, we have separated the notion of identity from liability, by suggesting that certificates explicitly specify liabilities needed by the party relying on a digital signature. We have illustrated this principle by introducing a pseudonym server that acts as an intermediate certificate issuer in existing certificate-based systems. The pseudonym certificates provide a relying party with clear guarantees, while the pseudonym server itself can calculate its own risk and minimize it through appropriate contracts with or deposits from requesting users.

We demonstrated the separation of identity from liability by applying it to the anonymization of existing transactions. It is, however, equally applicable in non-anonymous systems, where the presence of a certified identity may cause relying parties to make incorrect assumptions about liability guarantees. Also in these cases, making certifiers' liabilities explicit may help clarify objective guarantees, avoid unexpected liability gaps, and enhance trust.

References

1. Bellare, M., Garay, J.A., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G., Waidner, M. iKP – A Family of Secure Electronic Payment Protocols. In: Proc. First USENIX Workshop on Electronic Commerce. USENIX Assoc., Berkeley (1995) 89-106.
2. Bellare, M., Garay, J., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G.,Van Herreweghen, E., Waidner, M. Design, Implementation and Deployment of the iKP Secure Electronic Payment System. IEEE J. Sel. Areas in Commun. **18**, April 2000 issue, in press.
3. Mastercard and Visa. SET Secure Electronic Transactions Protocol, Version 1.0. Book One: Business Specifications; Book Two: Technical Specification; Book Three: Formal Protocol Definition. May 1997. Available from `http://www.setco.org/download.html`.
4. Asokan, N., Van Herreweghen, E., Steiner, M. Towards a Framework for Handling Disputes in Payment Systems. In: Proc. 3rd USENIX Workshop on Electronic Commerce, Boston, MA. USENIX Assoc., Berkeley (1998) 187-202.
5. The Anonymizer. `http://www.anonymizer.com`.
6. Gülçü, C., Tsudik, G. Mixing e-mail With Babel. In: Proc. 1996 Symposium on Network and Distributed System Security. IEEE Society Press, Los Alamitos (1996) 2-16.
7. Pfitzmann, A., Pfitzmann, B., Waidner, M. ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead. In: CI/ITC Conf.: Communication in Distributed Systems, Mannheim, Germany, February 1991. Informatik-Fachberichte 267. Springer-Verlag, Heidelberg (1991) 451-463.
8. ISO/IEC 9594-8 (X.509): OSI - The Directory - Authentication Framework.

9. Chaum, D. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. Commun. ACM **28** (1985), No. 10.
10. Lysyanskaya, A., Rivest, R., Sahai, A. Pseudonym Systems. Master's Thesis, MIT Laboratory for Computer Science (1999).
11. Chaum, D., Fiat, A., Naor, M. Untraceable Electronic Cash. In: Advances in Cryptology – Eurocrypt'89. Springer-Verlag, Berlin (1989) 319-327