Wallet Databases with Observers

(Extended Abstract)

David Chaum
CWI
The Netherlands

Torben Pryds Pedersen*
Aarhus University
Denmark

Abstract

Previously there have been essentially only two models for computers that people can use to handle ordinary consumer transactions: (1) the tamper-proof module, such as a smart card, that the person cannot modify or probe; and (2) the personal workstation whose inner working is totally under control of the individual. The first part of this article argues that a particular combination of these two kinds of mechanism can overcome the limitations of each alone, providing both security and correctness for organizations as well as privacy and even anonymity for individuals.

Then it is shown how this combined device. called a wallet, can carry a database containing personal information. The construction presented ensures that no single part of the device (i.e. neither the tamper-proof part nor the workstation) can learn the contents of the database — this information can only be recovered by the two parts together.

1 Introduction

In this paper we shall be concerned with a general system consisting of a number of individuals and organizations. Each individual has a small database with (personal) information (for example credentials), and the purpose of a transaction is to either update this database (obtain a new credential) or read some information in it (show a credential). For such a system it is important that the data in the database are correct:

- The organizations want to be sure that the contents of each database corresponds to what they have written in it.
- The individuals want to be sure that the organizations only store correct information in the database. and that they can only read and update those parts of the database that they are entitled to.

*Research partly done while visiting CWI

E.F. Brickell (Ed.): Advances in Cryptology - CRYPTO '92, LNCS 740, pp. 89-105, 1993.

Another basic requirement is that it should be possible for the individuals to participate anonymously in certain transactions. If, for example, the database contains medical information, which is needed in an investigation of a particular disease, the owner might require anonymity in order to participate in this investigation. There are, however, many other kinds of transactions, such as financial transactions, in which the issue of privacy is essential as well.

Section 2 argues that the *electronic wallet* is well suited for this scenario. An electronic wallet consists of two parts:

- A small, hand-held computer controlled by the user—denoted by C, for "computer"; and
- A tamper-proof module issued by the organizations—denoted by T, for "tamper-proof".

These two parts are arranged in such a way that T can only talk with C and not the outside world. This might be achieved by embedding T inside C. All communication with organizations is via C. It is essential that there is no "alternative way" that T can send messages to or receive messages from the outside world.

In the second part of the paper practical protocols are presented. First a new blind signature technique is presented in Section 3. Then Section 4 shows how, using the blind signatures, T can get a certified public key, which it can use to sign messages and thereby authenticate the actions taken by C. Then Section 5 presents the database protocols. In particular, it is shown how T can validate the information sent from the wallet without even knowing the contents of the database.

2 Possible Settings

This section discusses the advantages and disadvantages of different devices for use by individuals in a system including users and organizations, as described above. We shall primarily be concerned with how well the various alternatives support the requirements of correctness and privacy.

2.1 Correctness and Privacy

Correctness basically means that the data stored in a person's database can only be read or updated by the organizations/individuals that have permission to do so (according to some initially agreed rules). Note that these rules could say that a person is not allowed to change (parts of) his own database, and they could even (in extreme situations) specify that the user may not read parts of the database.

The terms *positive credential* and *negative credential* will be used to denote information in the database, which is to the advantage and disadvantage of the person, respectively. A bad criminal record is an example of a negative credential. The user may want to delete negative credential in the database, but this should, of course, be infeasible.

By *one-show credential* we mean a credential that the individual is allowed to show only once. Electronic money, which may be spent only once, is a typical example of a one-show credential.

While correctness is the most important requirement for organizations, privacy might be the important issue for individuals (at least in some situations), and it is essential for general acceptance of the system. We distinguish three levels of privacy:

- *Pure trust*:
  Information about the individual may be revealed during a transaction—the individual cannot do anything to enhance his privacy, but must trust the organization to maintain it.

- *Computational privacy*:
  If the individual follows the prescribed protocols, the organization cannot learn anything about him unless it can make a computation assumed to be infeasible.

- *Unconditional privacy*:
  If the individual follows the prescribed protocols, even an all powerful organization cannot learn anything extra about him.

2.2 Possible Approaches

We now analyze how two very different devices meet the demands of correctness and privacy outlined above. We first consider a device trusted completely by the individuals and then a tamper-proof device issued by the organizations (or an issuing center trusted by the organizations). This analysis then leads to the definition of electronic wallets.

Computer alone

First consider the situation where the user just has a computer, which he controls completely. In particular, he can delete or change any part of the memory, and he determines all messages which the device sends to the outside world.

Using the techniques of [Cha84] and [CFN90] it is possible to obtain unconditional privacy in this scenario in an efficient way. However, this setting makes it very difficult for the organizations to prevent users from deleting negative credentials or using one-show credentials more than once.

For example, in the case of an off-line electronic payments system, it is only known how to catch cheaters, who spend copies of the same electronic coin more than once, "after the fact". This method furthermore requires a large central database in which all valid coins are collected and compared (see [CFN90]), but this only has to be done periodically.

In short, this setting can give unconditional privacy, whereas no really efficient method for correctness is known.

Tamper-proof only

In this setting each individual has a tamper-proof module (packaged as a smart-card, for instance) issued by the organizations. Hence the organizations trust the correctness of the messages sent by the card, whereas the user does not even know which messages are being sent.

This approach gives correctness quite easily, because the tamper-proof part has to be broken in order to compromise the system. Furthermore, if cryptographic techniques are added it is sometimes possible to make systems in which cheating requires breaking the tamper resistant part *and* the cryptographic methods. Off-line electronic cash is an example of such a system.

If a tamper-proof unit is used to store negative credentials, the owner can delete these by destroying or throwing away the card. However, in this way he will also delete the positive credentials and, furthermore, the organizations will detect it the next time they need the card. In order to recover from such intentional as well as accidental losses of credentials, the system can have a back up facility for recovering such lost credentials. Hence, this approach can provide a very high degree of correctness.

The disadvantage of using the tamper-proof unit alone is that it only provides a low level of privacy, as the user has no control over the messages sent from the card. Therefore the card can (in principle) send any message that it likes during a transaction (e.g. the identity of the user). Hence, it can only give pure trust.

Electronic Wallets

The above analysis of two extreme settings shows that neither a user controlled computer nor a tamper-proof device alone can give sufficiently efficient and secure solutions. Electronic wallets can be thought of as a way to obtain the benefits of both approaches by a suitable combination.

Since no device that allows a tamper-proof device to communicate directly with the organizations can give a higher level of privacy than pure trust, the device must be constructed in such a way that the tamper-proof device cannot send messages to the organizations.

Thus the device should consist of a user controlled computer, C, with a tamper-proof unit, T, (sometimes called an *observer*), which on behalf of the organizations ensures that C cannot deviate from the prescribed protocols or change any information in its database. The electronic wallet is the simplest such device as it only has a single such observer (T). It might be useful (for example in order to make fault recovery easier) to have more than one observer, but such an approach does not seem to add significantly more power to the wallet.

Note, that C can freely communicate with the outside world without the knowledge of T, but the honest organizations will only accept messages which are approved by T.

The rest of this paper presents protocols, which show how T can control the actions of C. The fact that T may not communicate directly with organizations means that these protocols must be secure against

- *Inflow*:
  No matter how T and the organization deviate from the prescribed protocol, if C follows the protocol, the organization cannot send any extra (subliminal) information to T.

- *Outflow*:
  No matter how T and the organization deviate from the prescribed protocol. if C follows the protocol, T cannot send any extra (subliminal) information to the organization.

This means that even if the organization places a malicious observer in the wallet, there is no way that it can send back any information about the owner.

If all protocols are secure against outflow, then the security against inflow is not that significant, because T cannot tell other organizations what it learns. However, if it is important that T does not reveal any secrets in case it is returned to the organizations, the protocols must be secure against inflow as well.

3 The Signature Scheme

This section presents the signature scheme which will be used in this paper. The notation is introduced, the basic signature scheme is described, and it is shown how it can be used in wallets. Then it is shown how to make blind signatures.

3.1 Notation

Let q be a prime. The protocols to be presented work for any group, G_q of order q. As an example of such a group we consider another prime, p, such that q divides $p-1$, and define G_q as the unique subgroup of $\mathbb{Z}_p^*$ of order q. The element $g \in G_q$ will always be a generator of G_q. It will be assumed that all parties know p, q and g.

The discrete logarithm of $h \in G_q$ with respect to g is denoted by $\log_g h$, and the number of bits of an integer, x, will be denoted $|x|$.

3.2 The Basic Scheme

This subsection presents the signature scheme which will be used in the following protocols.

The public key of the scheme is

$$(p, q, g, h),$$

where $h \in G_q \setminus \{1\}$ and the corresponding secret key is $x = \log_g h$.

Let $m \in G_q$ be a message. The signature on m consists of $z = m^x$ plus a proof that

$$\log_g h = \log_m z.$$

Given m and z, consider the following protocol:

1. The prover chooses $s \in \mathbb{Z}_q$ at random and computes $(a, b) = (g^s, m^s)$. This pair is sent to the verifier.

2. The verifier chooses a random challenge $c \in \mathbb{Z}_q$ and sends it to the prover.

3. The prover sends back $r = s + cx$.

4. The verifier accepts the proof if

$$g^r = ah^c \qquad \text{and} \qquad m^r = bz^c.$$

If the prover can send correct responses r_1 and r_2 to two different challenges, c_1 and c_2 then

$$g^{r_1 - r_2} = h^{c_1 - c_2} \quad \text{and} \quad m^{r_1 - r_2} = z^{c_1 - c_2},$$

and hence

$$\log_g h = \log_m z = \frac{c_1 - c_2}{r_1 - r_2}$$

since $c_1 \neq c_2 \bmod q$ implies that $r_1 \neq r_2 \bmod q$. Now let H be a one-way hash function (as in the Fiat-Shamir scheme, see [FS87]). Given this function and the above protocol the signature on m is

$$\sigma(m) = (z, a, b, r).$$

It is correct if $c = H(m, z, a, b)$ and

$$g^r = ah^c \qquad \text{and} \qquad m^r = bz^c.$$

Hence, a signature on a $|q|$ bits message is $|q| + 3|p|$ bits long.

Now consider attempts to forge signatures given only the public key. If H has the property that it is as difficult to convince a verifier, who chooses $c := H(m, z, a, b)$, as a verifier who chooses the challenge at random (H is like a random oracle), it is not feasible to make signatures without knowing x.

Furthermore, it does not seem to help a forger to execute the proof that $\log_g h = \log_m z$ with the signer for the following reason. Consider the modification of the proof system in which the challenge, c, is chosen from a subset $A \subseteq \mathbb{Z}_q$ instead of $\mathbb{Z}_q$. For any such subset an execution of this modified scheme can be simulated perfectly in expected time $O(|A|)$. In particular this simulation is feasible if $|A|$ is polynomial in $|q|$. It is an open question to prove that executions of the protocol are secure, when A equals $\mathbb{Z}_q$, but we conjecture that no matter which $c \in \mathbb{Z}_q$ is chosen as challenge, the signer reveals no other information than the fact that $\log_g h$ equals $\log_m z$.

Finally remains the possibility that a forger can construct a false signature by combining various given signatures (m_i, σ_i), where the forger has chosen m_i adaptively (see [GMR88]). If $z_i = m_i^x$ then

$$z_1 z_2 = (m_1 m_2)^x.$$

Hence there is a multiplicative relation which might be useful for a forger. However, the use of H should prevent the forger from combining different signatures into a new signature.

3.3 Signatures by T

This section shows how the above signature scheme can be used by the tamper-proof device T in a wallet. The problem, that we have to deal with, is that T cannot be allowed to choose a and b alone, as it can encode some information in these two numbers. We therefore generate these two numbers using a coin-flipping protocol. If T has a public key (p, q, g, h_T) and a corresponding secret key $x_T = \log_g h_T$ it can sign a message $m \in G_q$ as follows:

1. C chooses $s_0 \in \mathbb{Z}_q$ and $t_0 \in \mathbb{Z}_q$ at random and sends $\alpha := g^{s_0} h_T^{t_0}$ to T (a commitment to s_0).

2. T chooses $s_1 \in \mathbb{Z}_q$ at random and sends $a_1 := g^{s_1}$ and $b_1 := m^{s_1}$ to C.

3. C sends (s_0, t_0) to T and computes $a := a_1 g^{s_0}$ and $b := b_1 m^{s_0}$.

4. T verifies that α equals $g^{s_0} h_T^{t_0}$ and computes $(a, b) := (a_1 g^{s_0}, b_1 m^{s_0})$.

5. T computes $c := H(m, m^{x_T}, a, b)$ and $r := s_0 + s_1 + c x_T \bmod q$.

The signature on m is (m^{x_T}, a, b, r).

It is not hard to see that if C follows the protocol then a and b are uniformly distributed in G_q. Furthermore, C can only open α as some $s_0' \neq s_0$ if it can find x_T. Hence, if T follows the protocol and C does not know x_T, then a and b are random elements of G_q.

Proposition 3.1
The above protocol for making signatures has the following two properties:

1. If C follows the protocol, then the signature is randomly distributed among the signatures on m — even if a cheating T has unlimited computing power.

2. If T follows the protocol, then a polynomially bounded cheating C learns no more than a random signature on m.

Proof
Both claims follow from the fact that the coin-flipping protocol in Step 1- 4 above has the following two properties:

1. (a, b) is uniformly distributed among the possible pairs, if C follows the protocol — even if a cheating T has unlimited computing power (because α contains no Shannon information about s_0).

2. A polynomially bounded C can only open α in two different ways if it knows $\log_g h_T$.

■

3.4 Blind Signatures

To get a blind signature on the message m in the above scheme one chooses a random $t \in \mathbb{Z}_q^*$ and asks the signer to sign $m_0 = m^t$. Let $z_0 = {m_0}^x$. Then the signer proves that $\log_g h = \log_{m_0} z_0$ in such a way that the messages are blinded:

1. The signer chooses $s \in \mathbb{Z}_q$ at random and computes $(a_0, b_0) = (g^s . {m_0}^s)$. This pair is sent to the verifier.

2. The verifier chooses $u \in \mathbb{Z}_q^*$ and $v \in \mathbb{Z}_q$ at random and computes

$$a = (a_0 g^v)^u \quad \text{and} \quad b = (b_0^{1/t} m^v)^u.$$

   (If both parties follow the protocol $a = (g^{s+v})^u$ and $b = (m^{s+v})^u$.) Then the verifier computes $z = z_0^{1/t}$, the challenge $c = H(m, z, a, b)$ and the blinded challenge $c_0 = c/u \bmod q$. The verifier sends c_0 to the signer.

3. The signer sends back $r_0 = s + c_0 x$.

4. The verifier accepts if

$$g^{r_0} = a_0 h^{c_0} \qquad \text{and} \qquad m_0{}^{r_0} = b_0 z_0^{c_0}.$$

   The verifier computes $r = (r_0 + v)u \bmod q$ and

$$\sigma = (z, a, b, r).$$

Proposition 3.2
σ is a correct signature on m, if the verifier accepts in the above protocol.

Proof
Let $c = H(m, z, a, b)$. We have to prove that

$$g^r = ah^c \quad \text{and} \quad m^r = bz^c.$$

The first equality follows from

$$g^r = (g^{r_0} g^v)^u = (a_0 h^{c_0} g^v)^u = (a_0 g^v)^u h^{c_0 u} = ah^c$$

and the second from

$$\begin{aligned} m^r &= m^{ur_0} m^{uv} \\ &= (m_0^{1/t})^{ur_0} m^{vu} \\ &= (m_0^{r_0})^{u/t} m^{vu} \\ &= (b_0 z_0^{c_0})^{u/t} m^{vu} \\ &= (b_0^{1/t} m^v)^u z_0^{uc_0/t} \\ &= bz^c. \end{aligned}$$

■

Proposition 3.3
The signer gets no information about m and σ if the receiver follows the protocol.

Proof

We will show that for all m, z, a, b and r such that

$$\begin{aligned} g^r &= ah^c \\ m^r &= bz^c \\ c &= H(m, z, a, b) \end{aligned}$$

and for all m_0, z_0, a_0, b_0, c_0 and r_0 such that

$$\begin{aligned} g^{r_0} &= a_0h^{c_0} \\ m_0^{r_0} &= b_0z_0^{c_0} \end{aligned}$$

there is exactly one set of values of t, u and v such that the signer sees $(m_0, z_0, a_0, b_0, c_0, r_0)$, when making the signature σ on m. In other words, that there is exactly one set of values of t, u and v such that

$$\begin{aligned} m &= m_0^{1/t} \\ a &= (a_0g^v)^u \\ b &= (b_0^{1/t}m^v)^u \\ c &= c_0u \\ r &= (r_0 + v)u. \end{aligned}$$

First, m and m_0 determine t as

$$m_0 = m^t \iff t = \log_m m_0.$$

Secondly, u and v are determined by c, c_0, r and r_0 as

$$u = \frac{c}{c_0} \quad \text{and} \quad v = \frac{c_0}{c}r - r_0.$$

Thus we just need to show that these values of t, u and v satisfy

$$a = (a_0g^v)^u \quad \text{and} \quad b = (b_0^{1/t}m^v)^u.$$

In doing this it can be assumed that $z_0 = m_0^x$ and $z = m^x$, because the signer actually proves that z_0 equals m_0^x when making a blind signature. Hence $m_0 = m^t$ implies that

$$z_0 = z^t.$$

The first equality is proven as follows

$$a = g^rh^{-c} = g^{(r_0+v)u}h^{-uc_0} = (g^{r_0}g^vh^{-c_0})^u = (a_0g^v)^u.$$

The second equality follows by similar rewritings:

$$\begin{aligned} b &= m^rz^{-c} \\ &= m^{(r_0+v)u}z^{-c_0u} \\ &= (m^{r_0}m^vz^{-c_0})^u \\ &= ((m_0^{1/t})^{r_0}m^v(z_0^{1/t})^{-c_0})^u \\ &= ((m_0^{r_0}z_0^{-c_0})^{1/t}m^v)^u \\ &= (b_0^{1/t}m^v)^u. \end{aligned}$$

This completes the proof. ∎

Hence, this signature scheme allows the receiver to obtain blind signatures. In particular it is possible for the receiver to get a signature on any message that he chooses. In order to avoid this problem in the application to wallets, the organization only signs a blinded message if the challenge is signed by T. The resulting scheme is presented in the next subsection.

3.5 Blind Signatures in Wallets

We assume that a center Z is the signer. The public key of Z is h_Z and the secret key is $x_Z = \log_g h_Z$.

1. C chooses the blinding factor $t \in \mathbb{Z}_q^*$ at random and sends $m_0 := m^t$ to Z.
2. Z and C choose a_0 and b_0 using a coin-flipping (as in Section 3.3) protocol, such that only Z knows $s = \log_g a_0 = \log_{m_0} b_0$.
3. Z computes $z_0 := m_0^{x_Z}$ and sends it to C.
4. C computes $z := z_0^{1/t}$ and chooses u and v at random. Then it sends (a_0, b_0, z, u, v, t) to T.
5. Both T and C can then compute $a := (a_0 g^v)^u$, $b := (b_0^{1/t} m^v)^u$, $c := H(m, z, a, b)$ and $c_0 := c/u$. T signs c_0 and sends it to C.
6. C verifies the signature before sending the challenge and the signature to Z.
7. From now on the protocol for constructing and verifying blind signatures is followed. Hence Z computes the response, r_0, and sends it to C. C verifies this response before forwarding it to T. Finally T unblinds r_0 and verifies the signature.

Theorem 3.4
If C follows the protocol then

1. Z gets no information about the signature on m.
2. T sends no information to Z except a random signature on c_0.
3. Z sends no information to T except z_0.

Proof
Assume that C follows the protocol.

1. Z sees messages with the same distribution as in the original protocol for making blind signatures — except that Z cannot choose (a_0, b_0) freely anymore. But this pair is chosen at random. Hence this property follows from Proposition 3.3.

2. The only information, which originates from T is the signature on c_0. However, Proposition 3.1 implies that this signature is randomly chosen among the possible signatures.

3. T sees the following messages from Z:

$$(a_0, b_0), z_0^{1/t} \text{ and } r_0,$$

and T receives u, v and t from C. Here (a_0, b_0) is uniformly distributed (by the same argument as in the proof of Proposition 3.1), and r_0 is uniquely determined. Hence, Z can only send information to T via z.

■

Note that if Z does not compute z_0 as $m_0^{x_Z}$ then C will discover it. Thus, it is impossible for Z to send information to T without being detected. However, as we shall see in the next section even this possibility of inflow is eliminated in our application of the protocol.

We now look at the security of the protocol and assume that T and Z both follow the protocol. It will be argued that if the basic signature scheme is secure, and if T's signatures cannot be faked, then no matter what a polynomially bounded $\tilde{C}$ does, it learns no more than a random signature on m.

As $\tilde{C}$ cannot forge T's signatures, it can be assumed that c_0 is computed as $c_0 := H(m, \tilde{z}, \tilde{a}, \tilde{b})$, where $\tilde{C}$ can choose $\tilde{z}$, $\tilde{a}$ and $\tilde{b}$, but not m. By the assumption about H this means that $\tilde{C}$ cannot control the value of c_0 ($\tilde{C}$ cannot force c_0 to be any particular value, except by trying different values for $\tilde{z}$, $\tilde{a}$ and $\tilde{b}$ and hoping they will give a "good" value of c_0). Thus $\tilde{C}$ does not seem be better off in this situation than when it just gets a "normal" signature from the signer.

4 Obtaining a Pseudonym

This section shows how the wallet can get a public key, which is signed by a key authentication center. The signature on the public key will be called a *validator*. This protocol has the property that neither the center nor any other unlimited powerful organization can link the identity of the user to the public key (or its validator).

Combining this result with Section 3.3 gives a method for T to sign messages without revealing any information at all about the owner of the wallet. This provides a method for T to validate the messages, which C sends to the outside world, without revealing anything about the identity of the user; these messages are only accepted by the organizations if they are signed properly (by T).

We now show how T can generate a secret key $x \in \mathbb{Z}_q^*$ and obtain a certificate on the corresponding public key $h = g^x \bmod p$. In order to get started, it is assumed that each T is born with a secret key, x_T, and a corresponding public key, h_T, to the signature scheme described in Section 3.3. These signatures can be traced to T (and hence to the individual), and they are therefore only used in an initial step where T gets a validated key from a key authentication center (Z). The center issues validators using the blind signature scheme from the previous section with secret key x_Z and public key h_Z.

The basic idea of the protocol for issuing validators is that C and T first execute a coin-flipping protocol in order to choose a secret key, x, which only T learns. The corresponding public key is denoted by h. Then C chooses a blinding factor, $t \in \mathbb{Z}_q^*$, and C signs the blinded public key $(h_1 = h^t)$. Note, that in the process of making the blind signature, T has to sign a challenge computed as $H(h, h^{x_Z}, a, b)$. This signature guarantees to Z that it validates a public key which is accepted by T. There is no need that T signs h_1 before Z starts making the blind signature, because before Z computes the response, it only produces random messages, which a cheating C could have produced by itself. In more detail the protocol goes like this:

1. C chooses $y_0 \in \mathbb{Z}_q^*$ at random and sends a commitment to y_0 to T.
2. T chooses $y_1 \in \mathbb{Z}_q^*$ at random and sends $h_0 := g^{y_1}$ to C.
3. C opens the commitment and sends y_0 to T.
4. T and C compute $h := h_0^{y_0}$, and T computes the secret key $x := y_0 y_1 \bmod q$.
5. T computes $z := h_Z^x$ and sends it to C.
6. C chooses $t \in \mathbb{Z}_q$ at random and sends $h_1 := h^t$ to Z.
7. Z makes a blind signature on h by signing h_1 as follows:
   (a) Z computes $z_0 := h_1^{x_Z}$. Then Z and C choose $(a_0, b_0) := (g^{s_0}, h_1^{s_0})$ at random such that only Z knows s_0, whereas both know a_0 and b_0. Z sends z_0 to C.

   (b) C first verifies that $z_0 = z^t$, and then it chooses $u \in \mathbb{Z}_q^*$ and $v \in \mathbb{Z}_q$ at random and computes
   $$a := (a_0 g^v)^u \qquad \text{and} \qquad b := (b_0^{1/t} h^v)^u.$$
   C then sends u, v, t and (a_0, b_0) to T.

   (c) T computes the pair (a, b) just as C did, the challenge $c := H(h, z, a, b)$, and $c_0 := c/u \bmod q$. Then it signs c_0 using x_T (with help from C) and sends the signature to C.

   (d) C computes $c := H(h, z, a, b)$, $c_0 := c/u$, and verifies the signature. C then forwards c_0 and the signature to Z.

   (e) Z verifies the signature on c_0 and computes $r_0 := s_0 + c_0 s_Z \bmod q$.

   (f) C verifies that
   $$g^{r_0} = a_0 h_Z^{c_0} \qquad \text{and} \qquad h_1^{r_0} = b_0 z_0^{c_0}$$
   and computes $r = (r_0 + v)u \bmod q$. Then C forwards r_0 to T.

   (g) T computes $r := (r_0 + v)u \bmod q$ and verifies that:
   $$g^r = a h_Z^c \qquad \text{and} \qquad h^r = b z^c.$$

Theorem 4.1
This protocol satisfies:

1. If T, C and Z follow the protocol, then T gets Z's signature on h.
2. If C follows the protocol then Z gets no information about h or σ. This is true even if T and Z have unlimited computing power.
3. If C follows the protocol then Z can construct all messages with the same distribution in expected polynomial time except the signature on c_0.
4. If C follows the protocol, then T can simulate all messages that it receives — except r_0.
5. If the blind signature scheme is secure then a polynomially bounded $\tilde{C}$ cannot get a validated public key for which he knows the corresponding secret key.

Proof
The first three properties are straightforward to prove, and the fourth follows from Theorem 3.4 and the fact that T can compute z by itself. As for the last property, note that the security of the blind signature scheme means that $\tilde{C}$ can only get a signature on h, but $\tilde{C}$ cannot find the secret key corresponding to h (i.e. $\log_g h$) unless it can compute discrete logarithms in G_q. ∎
As C can make sure that the signature on c_0 is random among all possible signatures, this theorem shows that the protocol for issuing a validated public key has no outflow. Furthermore, as r_0 is uniquely determined from the other messages the protocol protects against inflow.

5 An Application to Databases

This section first describes how a very simple database offering unconditional privacy as well as correctness can be constructed, and then it is shown how a database in which the information is kept secret from both T and C can be constructed. By similar techniques, it is also possible to construct databases in which

1. The data is known by T, but kept secret from C; and
2. The data is known by C, but kept secret from T.

Whenever T signs a message (anonymously) with respect to a public key, which is validated by the key authentication center, the signature will be referred to as a certified signature.

5.1 A Simple Database

The wallet can be used to store the personal database described in the introduction as follows:

- All information in the database is stored by T and C.

- Whenever an organization updates a field in the database, it sends a signed message to the wallet. C verifies the signature before it updates the database and forwards the new information plus the signature to T. Finally T verifies the signature and updates the database.

- When an organization wants to read a field in the database (or a function-value of several fields), a certified signature on the value is sent to the organization.

5.2 Database with Hidden Information

The implementation presented above has the property that both T and C know all information in the database. This could be a little dangerous for the user, because T could leak all information, in case it is captured by another person, who is able to break the tamper-resistance. On the other hand, there might be certain very sensitive data in the database, which the user should not know either (or does not want to be stored in his computer).

In the following it is therefore shown how the above database can be modified such that neither T nor C knows the data, but T is still able to control that C does not change anything in the database. We shall, however, only give protocols which allow the organization to read or write a single bit in the database. The following scheme for probabilistic encryption is an important ingredient in these protocols.

Probabilistic Encryption

Let $n = pq$, where p and q are primes both equivalent to 3 modulo 4. In order to encrypt a bit b, the committer chooses $r \in \mathbb{Z}_n^*$ at random and computes

$$BC(n, b, r) := (-1)^b r^2 \bmod n.$$

A person knowing p and q can decipher a given ciphertext by determining whether it is a quadratic residue or not. However, for a person not knowing p and q this is presumably infeasible.

Let n_1 and n_2 be two different moduli as above, and let $\beta_1 = (-1)^b r_1^2 \bmod n_1$ and $\beta_2 = (-1)^b r_2^2 \bmod n_2$ be probabilistic encryptions of the same bit $b \in \{0, 1\}$.

Theorem 5.1

There exists a four-round protocol with security parameter k in which a person, P, knowing r_1 and r_2 can prove to another person, V, that β_1 and β_2 are in fact encryptions of the same bit. More precisely this protocol satisfies:

1. If P and V follow the protocol, then V will accept, if $a = b$.

2. If V follows the protocol and $a \neq b$, then V will reject the proof with probability at least 2^{-k} no matter what an unlimited powerful prover does.

3. It is a proof of knowledge of r_1 and r_2.

4. It is (computational) witness hiding (see [FS90]).

Proof

The protocol uses the cut-and-choose technique. The details are omitted here. ■

The Protocols

It is assumed that each organization, W, has a modulus, n_W, as above, and that W can make digital signatures. Prior to the execution of the read and write protocols to be described, the following start-up protocol is executed:

1. W constructs a request of the form $(n_W, op, name, time)$, where $op \in \{read, write\}$, $name$ identifies the bit which W wants to read or write, and $time$ is a time-stamp. This request is signed and sent to the wallet together with certificates, which show that n_W is a valid modulus and that the public key of W (for the signature scheme) is valid.

2. C verifies the request and certificates, and if they are legal, C forwards them to T. In particular, C verifies that $time$ is constructed correctly so that W has not encoded any information in it.

3. T verifies the request and the certificates.

Whenever T and C sign a message in the certified signature scheme op, $name$, $time$ and n_W are included in the message. This prevents obvious frauds by C in which signatures from previous executions of the same or different protocols are reused.

Furthermore, each write protocol must be immediately followed by a protocol in which T sends a signed message to W (through C) in which it confirms having received the required messages.

For each bit b in the database, T has given C a commitment $\beta_T = BC(n_0, b_T, r_T)$ to a bit b_T, and C has given T a commitment $\beta_C = BC(n_0, b_C, r_C)$ to a bit b_C such that $b = b_T \oplus b_C$. The modulus n_0 is the modulus of the organization which wrote b. An organization, W, with public modulus n_W can read b as follows

1. T chooses $s_T \in \mathbb{Z}^*_{n_W}$ at random and sends $\alpha_T := (-1)^{b_T} s_T^2 \bmod n_W$ to C.
   T proves to C that α_T and β_T are encryptions of the same bit.

2. C chooses $s_C \in \mathbb{Z}^*_{n_W}$ at random and sends $\alpha_C := (-1)^{b_C} s_C^2 \bmod n_W$ to T.
   C proves to T that α_C and β_C encrypt the same bit.

3. T and C sign $\alpha := \alpha_T \alpha_C$ using the certified signature scheme.
   This signature (and α) is sent to W (through C).

4. W verifies the signature and finds the encrypted bit by deciphering α.

This protocol has the following properties:

- If C follows the protocol: No matter what (an unlimited powerful) T does, α is a random encryption of b. Furthermore, the signature on α does not contain any information other than the fact that a legal T produced it.

- If T follows the protocol, then α is an encryption of b as long as C cannot fake T's signatures (or break the tamper-proofness).

- It does not make it easier for T and/or C to find b unless W tells them how to distinguish encryptions of 0 from encryptions of 1 modulo n_W.

The proofs of these properties are quite straightforward, and they are omitted from this extended abstract. The organization, W, can write a bit, b, in a given field in the database as follows:

1. T chooses $a_T \in \{0, 1\}$ and $r_T \in \mathbb{Z}_{nw}^*$ at random and sends $\alpha_T := (-1)^{a_T} r_T^2 \bmod n_W$ to C.

2. C chooses $a_C \in \{0, 1\}$ and $s_C \in \mathbb{Z}_{nw}^*$ at random and sends $\alpha_C := (-1)^{a_C} s_C^2 \bmod n_W$ to T.

3. T and C sign $\alpha := \alpha_T \alpha_C$ in the certified signature scheme. This signature (and α) is sent to W (through C).

4. W verifies the signature and finds the bit a by deciphering α.

5. W and C choose $r \in \mathbb{Z}_{nw}$ at random using a coin-flipping protocol.

6. W then computes $b' = a \oplus b$ and $\alpha_W := (-1)^{b'} \alpha r^2$, which it subsequently signs. W sends the signature (σ_W) to C.

7. C computes α_W, verifies σ_W and computes $\beta_C := \alpha_W \alpha_T^{-1}$ and $\beta_T := \alpha_T$ and $r_C := r s_C$ and
$$b_C := \begin{cases} a_C & \text{if } \alpha_W = \alpha r^2 \\ a_C \oplus 1 & \text{if } \alpha_W = -\alpha r^2. \end{cases}$$
C then forwards α_W and σ_W to T.

8. T verifies the signature and computes $\beta_C := \alpha_W \alpha_T^{-1}$ and $\beta_T := \alpha_T$ and lets $b_T = a_T$.

This protocol satisfies (again the proofs are omitted):

- If T, C and W follow the protocol then after the execution the following holds:
    1. $\beta_T = BC(n_W, b_T, r_T)$;
    2. $\beta_C = BC(n_W, b_C, r_C)$;
    3. $b = b_T \oplus b_C$.

- If C cannot fake T's or W's signatures then $b \oplus b_T$ equals the plaintext corresponding to β_C.

- After the execution $b \oplus b_C$ equals the plaintext corresponding to β_T no matter what an unlimited powerful T does.

- C and/or T can only find b if they can distinguish quadratic residues from quadratic non-residues modulo n_W.

- If C follows the protocol, then W just gets a signature on a random encryption of a random bit. Similarly, T just gets a random encryption of a random bit chosen by W.

In the above two protocols the amount of inflow and outflow is very limited. Note, that W could have told T the factorization of n_W in advance. Hence, T learns the bit. However, this does seem to be a serious problem as W already knows this bit.

6 Conclusion and Future Work

We have argued that the electronic wallets presented here are an excellent way to store personal databases. And we have shown protocols that allow T to control and validate all messages from the user to the outside world. These protocols allow C to ensure that the privacy of the person is not compromised. They provide organizations with security against abuse by individuals that relies on the assumption that the tamper-proofness cannot be broken and that the signatures cannot be forged.

The protocols presented do, however, have a limited kind of inflow because T and W see the same random values (such as those used to form the signatures). In case T gets captured, these values would let organizations who could read out the contents of a captured T link it to specific protocol instances. Forthcoming joint work with Stefan Brands, Ronald Cramer and Niels Ferguson shows how the need for observers and organizations to share such information can be avoided altogether.

References

[CFN90] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology - proceedings of CRYPTO 88*, Lecture Notes in Computer Science. pages 319 - 327. Springer-Verlag, 1990.

[Cha84] D. Chaum. Blind signature systems. In *Advances in Cryptology - proceedings of CRYPTO 83*, 1984.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - proceedings of EUROCRYPT 86*, Lecture Notes in Computer Science, pages 186 - 194. Springer-Verlag, 1987.

[FS90] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 416 - 426, 1990.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen message attack. *SIAM Journal on Computing*, 17(2):281 - 308, April 1988.