INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Blinded-Key Signatures: securing private keys embedded in mobile agents

Lucas de Carvalho Ferreira   *Ricardo Dahab*

Technical Report - IC-01-15 - Relatório Técnico

November - 2001 - Novembro

Blinded-Key Signatures: securing private keys embedded in mobile agents

Lucas de Carvalho Ferreira Ricardo Dahab[*]

Abstract

In this paper we present a new cryptographic primitive, the *blinded-key signature*, that allows the inclusion of private keys in autonomous mobile agents. This novel signature scheme can be applied to several well known digital signature schemes, such as RSA and ElGamal.

1 Introduction

Mobile agent computing is an increasingly important paradigm, which presents interesting new security challenges [Chess, 1998]. Among these are the protection of hosts against malicious agents and the protection of agents against malicious hosts. While many solutions have already been proposed and implemented for protecting hosts that execute agents, it appears to be more difficult to find practical solutions to the problem of protecting agents.

Specially of interest is the use of agents in electronic commerce settings, in which users release autonomous agents to roam around virtual stores and find desired products at acceptable prices. In these environments it is desirable to automate the purchasing process, allowing agents to perform tasks such as selection of items and issuing and signing payment orders. Naturally, this poses a security threat to the private keys carried by such autonomous agents. Traditional proxy signatures [Mambo et al., 1996] provide a solution to this problem. Here we propose a new proxy signature scheme, the *blinded-key signature*, which allows users to protect private keys embedded in mobile agents, while allowing agents to securely sign documents on behalf of their owners.

Blinded-keys consist of private keys that have been *hidden* in such a way as to allow the execution of the signature generation algorithm in an untrusted environment while keeping the original private key secret. Signature verification can be executed with the help of the information used to blind the key. This technique can yield powerful results when used together with data hiding techniques, especially *time-limited black boxes*[Hohl, 1998].

In Section 2, we present the architecture of our system, i.e., the main entities and interactions among them. Section 3 describes the algorithms for blinding signature keys and discusses security issues and Section 4 discusses policies that can be used in blinded-key signature systems. Section 5 extends the blinded-key signature framework to better protect

[*]Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP, Brazil

Figure 1: Entities of a blinded-key signature scenario.

user privacy. Related work and the complementary technique of time-limited black boxes are described in Section 6 and a use example of blinded-keys is shown in Section 7; Section 8 concludes the paper.

2 The basic architecture

In this section, we give an overall description of the basic blinded-key signature scenario. We first describe the entities and then their interaction. This description fits most models for mobile agents, in that someone builds agents and releases them on a network in order to achieve some specified goal. Agents are autonomous in the sense that they do not need to contact their owners after they are released: they carry algorithms and rules that allow them to act on their owner's behalf.

The entities

We deal with four different entities, as shown in Figure 1:

Owner: the owner is the entity which builds and configures the agents and specifies their mission, releasing them on the network to accomplish their job. The owner has a pair of public/private keys that may be certified by a Certification Authority.

Agents: are mobile pieces of code that will execute some action on behalf of their owners. An agent has a unique ID and carries a blinded key for signing agreements. If an agent signs an agreement, its owner will be bound by its terms.

Foreign Agency: an agency is a place where agents can bind to and execute their code. The foreign agency may be an agent-based marketplace, some entry point to a distributed database, etc. The foreign agency is not trusted by the owner and may try to disrupt an agent's behavior or have access to the agent's internal state.

Notary: is a Trusted Third Party (TTP) that will be responsible for verifying blinded-key signatures and enforcing agreements signed by agents. The notary may also certify owners' public keys or execute other TTP tasks such as verifying owner payment records.

The interactions

We show how these architectural entities interact by means of an example. Suppose Alice (owner) wants to send an agent to a virtual store (foreign agency) with the purpose of contracting the acquisition of a copy of the original RSA paper. The following steps are executed:

1. In order to prevent the store from having access to its private key, Alice blinds it, using a *blinding factor*, before including it in the agent. Alice also sets the agent's mission and specifies the maximum price of \$10.
2. Alice sends the notary a message containing a signed *policy* describing the kinds of negotiations and contracts this agent is authorized to sign. Section 4 discusses items that may figure in the policies. The policy itself must include both the agent's ID and the blinding factor. This step and the next are pictured in Figure 1.
3. Alice releases her agent which will interact with the store in order to locate the RSA paper and to negotiate its price.
4. The agent signs a purchase or payment order for the paper using the *blinded private key*.
5. The store contacts the notary and asks it to verify the signature and the purchase or payment order with respect to the policy specified by Alice for this agent. The store accepts the order if the notary correctly verifies the signature on it and the deal conforms to the owner's policy.
6. The store notifies the agent and releases it to its next stop.

3 Blinded-key signatures

To prevent agencies from having access to their private keys, owners will blind them before inclusion in their agents. In this section, we first present the blinded-key signature scheme for the RSA system and then we generalize it to other signature methods. From this point to the end of the paper, we consider messages and signatures to be integers in the interval $[0, n-1]$, where n is the modulus used in the signature scheme.

3.1 Blinded-key signatures using RSA

Recall that the public and private keys of an RSA user are, respectively, (e, n) and (d, n) (or just d), where: (i) $n = pq$, for p and q large prime numbers; (ii) e and d are integers such that $ed \equiv 1 \pmod{\phi(n)}$, for $\phi(n) = (p-1)(q-1)$ and $1 \leq e, d \leq \phi(n) - 1$. More about the RSA can be found in [Rivest et al., 1978, Menezes et al., 1997].

Blinding the key. To blind an RSA private exponent d, the owner executes Algorithm 1, which produces the private blinded exponent d_b. The blinded private key consists of the pair (d_b, n) (or just d_b).

Signing with the blinded key. Signature generation with blinded keys uses exactly the same algorithm as signing with standard RSA keys. Given a message M and a blinded private key (d_b, n), a signature S is simply

$$S := M^{d_b} \bmod n.$$

Algorithm 1 Blinding an RSA private key.

Given the private exponent d and $\phi(n)$, compute the private blinded exponent d_b as follows:

1. Choose a random blinding factor b, an integer in $[0, \phi(n) - 1]$.

2. The blinded private exponent d_b is defined as

$$d_b := db \bmod \phi(n).$$

Verifying a blinded-key signature. To verify a blinded-key signature S, the notary executes Algorithm 3.

Algorithm 2 Origibal RSA signature verification.

Given a message M, the corresponding signature S, the signer's public key (e, n),

1. compute $S_1 :=\mathrm{M}^e \bmod n$;

2. Accept the signature if and only if $S = S_1$.

Algorithm 3 RSA blinded-key signature verification.

Given a message M, the corresponding signature S, the signer's public key (e, n) and the private key's blinding factor b,

1. compute $S_1 := S^e \bmod n$ and $S_2 := M^b \bmod n$;

2. Accept the signature if and only if $S_1 = S_2$.

Note that the verification process is correct since

$$S_1 \equiv S^e \equiv M^{d_b e} \equiv M^{dbe} \equiv M^b \equiv S_2 \pmod n$$

.

3.2 Generalized blinded-key signatures

We now generalize the blinded-key signatures and sketch how this scheme works for several other signature methods.

Basic assumptions. To be able to generalize blinded-key operations to other cryptosystems, we assume that the underlying signature method is based on arithmetic in cyclic groups of order n.

Blinding the key. The general process for blinding the key generalizes to the process used for the RSA. In the lack of a secret modulus such as $\phi(n)$, we use n as the modulus for the blinding process, which is defined in Algorithm 4.

Algorithm 4 General key-blinding process

Given the private key a and a modulus n, compute the blinded key as follows:

1. Choose at random a blinding factor $b \in Z/nZ$.
2. The blinded key a_b corresponding to a and y is defined as:

$$a_b \equiv ab \bmod n.$$

Simply put, blinding a key means choosing a random blinding factor and computing the product of the private key and the blinding factor modulo n. If the private key is composed of many parts, as in the Feige-Fiat-Shamir signature scheme (see Section 3.2.1), the blinding factor will also consist of many parts and each part of the blinded key corresponds to the modular product of one key part with one of the parts of the blinding factor.

Signing with the blinded key. The signature operation for blinded-key signatures consists of applying the same algorithm used for the original signature scheme using the blinded key instead of the private key, as we have already shown for the RSA scheme.

Verifying a blinded-key signature. The signature verification step is highly dependent of the underlying signature scheme, and we will not try to define it as a general algorithm. Rather, we describe in algorithm 5 how this step can be derived from the corresponding step for the underlying signature scheme.

Algorithm 5 Defining the signature verification step for blinded keys.

Given a signature S and an algorithm $f(a, S)$ which defines the verification step for the underlying signature scheme, the signature verification algorithm $f_b(a_b, S)$ for blinded keys is defined as:

1. Define $f(a, S)$ as consisting of two steps: a computation $g(a, S) = r$ and a test $h(r, S)$, with the output of the verification algorithm f being the result of the test h.
2. Compute $t = g(a_b, S)$. If the signature scheme being adapted is suited to be used with blinded keys, t should be defined in terms of b, S, and of public information (general parameters of the signature scheme and the corresponding public key).
3. Define a new test function $h_b(t, S, b)$ that outputs *true* if and only if $h(r, S)$ would output true.

3.2.1 Examples for other signature schemes.

We now sketch how some other signature schemes may be adapted to the blinded-key signature framework. We do not detail all operations, just the signature verification step. The other steps for those signature schemes can easily be derived by the algorithms shown above. For more detail on the signature schemes used in this section, the reader is referred to [Menezes et al., 1997].

ElGamal: In the ElGamal system, the private key is denoted a and the blinded key $a_b = a \times b \bmod (p-2)$, where $b \in [0, p-2]$ is the blinding factor. The signature is composed of two parts and denoted (r, s). The verification step is shown in algorithm 7.

Algorithm 6 Original ElGamal signature verification

Given the siganture (r, s) and the public key (p, α, y),

1. compute $v_1 = y^r r^s \bmod p$ and $v_2 = \alpha^{h(m)} \bmod p$;
2. Accept the signature if and only if $v_1 = v_2$.

Algorithm 7 ElGamal verification step for blinded-key signatures.

Given the signature (r, s) and a public key (p, α, y), the computation step is

$$\begin{aligned} v_1 &= y^{br} r^s \bmod p \\ v_2 &= \alpha^{h(m)} \bmod p \end{aligned}$$

where $h(m)$ is a hash of the message that was signed.
The test for this signature is

$$v_1 \overset{?}{=} v_2.$$

Digital Signature Standard (DSA) The US Digital Signature Standard (DSA) is a variant of the ElGamal system, in which the private key is also denoted a and the signature (r, s). We will also denote the blinded key $a_b = a \times b \bmod (q-1), b \in [0, q-1]$. The verification step is slightly different, as shown in algorithm 9.

Schnorr Schnorr signatures are another variant of the basic ElGamal Scheme. The private key is denoted a, the signature is (s, e) and the blinded key is $a_b = a \times b \bmod q, b \in Z_q$. The verification step is shown in algorithm 11.

Feige-Fiat-Shamir This is an example of a signature scheme where the keys are composed of several parts and this makes the blinding factor the be composed of many parts also. For instance, the private key is composed of k parts, where k is the number of bits of the output

Algorithm 8 Original DSA verification step

Given a signature (r, s) and a public key (p, q, α, y),

1. compute $w = s^-1 \bmod q$,
2. compute $u_1 = w.h(m) \bmod q$ and $u_2 = rw \bmod q$,
3. compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$,
4. Accept the signature if and only if $v = r$.

Algorithm 9 DSA verification step for blinded-key signatures.

Given a signature (r, s) and a public key $(p, q, \alpha\, y)$, the computation step is

$$\begin{aligned} w &= s^{-1} \bmod q \\ u_1 &= w.h(m) \bmod q \\ u_2 &= rw \bmod q \\ v &= (\alpha^{u_1} y^{bu_2} \bmod p) \bmod q \end{aligned}$$

where $h(m)$ is a hash of the message being verified.
The test for this signature scheme is

$$v \overset{?}{=} r.$$

Algorithm 10 Original Schnorr verification step

Given a signature (s, e) and a public key (p, q, α, y), execute the following steps:

1. compute $v = \alpha^s y^{-e} \bmod p$ and $e' = h(m||v)$;
2. Accept the signature if and only if $e' = e$.

Algorithm 11 Schnorr verification step for blinded-key signatures

Given a signature (s, e) and a public key (p, q, α, y), the computation step is

$$\begin{aligned} v &= \alpha^s y^{-eb} \bmod p \\ e' &= h(m||v) \end{aligned}$$

where $h(m)$ is a hash of the message.
The test step for this signature scheme is

$$e' \overset{?}{=} e.$$

of the hash function to be used while signing, and is denoted $(s_1, s_2, \ldots, s_k)$, the blinding factor will be composed of k different random parts $(b_1, b_2, \ldots, b_k)$ and the blinded key will be $(s_{b_1}, s_{b_2}, \ldots, s_{b_k})$ for $s_{b_i} = s_i \times b_i \bmod n$ for $i = 1, 2, \ldots, k$ and $b_i \in Z_n$. The signature is composed of two parts, namely (e, s). Algorithm 13 shows the adapted verification step for this signature scheme.

Algorithm 12 Original Feige-Fiat-Shamir verification procedure

Given a signature (e, s) and a public key $(v_1, v_2, \ldots, v_k)$, execute the following steps:

1. Compute $w = s^2 . \prod_{j=1}^{k} v_j^{e_j} \bmod n$;
2. Compute $e' = h(m||w)$;
3. Accept the signature if and only if $e = e'$.

Algorithm 13 Feige-Fiat-Shamir verification step for blinded-key signatures.

Given a signature (e, s) and a public key $(v_1, v_2, \ldots, v_k)$, the computation step is

$$\begin{aligned} w &= s^2 \prod_{j=1}^{k} v_j^{e_j} \bmod n \\ z &= \prod_{j=1}^{k} b_j^{2e_j} \\ e' &= h(m||(w.z^{-1})) \end{aligned}$$

where $h(\cdot)$ is a hash function whose output is k bits long, e_j is the j-th bit of e and m is the message which was signed.
The test step for this signature scheme is

$$e \stackrel{?}{=} e'.$$

Guillou-Quisquater As the Feige-Fiat-Shamir, this is another example of signature schemes based on identification protocols. For this scheme, the private key is denoted a and the signature is composed of two parts, namely (s, e). The blinded key is defined as $a_b = a \times b \bmod n, b \in Z_n$. The verification step for this signature scheme is defined in Algorithm 14.

3.3 Security of blinded-key signatures

First, let us assume that the notary is trusted and does not misbehave. In this case, the main threat comes from the foreign agency trying to access the agent's internal state and possibly acquiring its blinded-key pair. This would allow the agency to sign agreements on

Given a signature (s, l) and a public key (n, e, J_A), execute the following steps:

1. Compute $u = s^e J_A^l \bmod n$ and $l' = h(m||u)$;
2. Accept the signature if and only if $l = l'$.

Algorithm 14 Guillou-Quisquater verification step for blinded-key signatures.

Given a signature (s, l) and a public key (n, e, J_A), the computation step is defined as

$$\begin{aligned} u &= s^e J_A^l y^{-e} \\ l' &= h(m||u) \end{aligned}$$

where $h(\cdot)$ is a hash function.
The test step for this signature scheme is

$$l' \overset{?}{=} l.$$

behalf of the agent's owner but still would not enable the agency to compute the owner's private key, as shown below. The risk to the owner posed by this threat is limited since the key is bound to a restricting policy at the notary. The notary only considers valid those blinded-key signatures that abide by the policy specified for the key. So, in any case, the agency can only use the blinded key to sign an agreement the agent could have negotiated himself.

Can the agency determine the owner's private key? The agency knows the blinded key $a_b = a \times b \bmod n$, where b and a are unknown. It is clear that solving this equation for y means guessing it, which is at least as difficult as guessing a itself. So we conclude that breaking the security of blinded-key signatures is, for the agency alone, at least as difficult as breaking the underlying cryptosystem by guessing a private key.

Another concern for blinded keys is if routine use of the scheme would convey malicious agencies enough information to determine the user's private key. For this matter, an agency or a collusion of agencies can collect a number of blinded keys

$$\begin{aligned} a_{b_1} &= a \times b_1 \\ a_{b_2} &= a \times b_2 \\ &\vdots \\ a_{b_n} &= a \times b_3 \end{aligned}$$

with $b_1 \neq b_2 \neq \ldots \neq b_n$. We will assume that n is large enough so the owner will never need to reuse a blinding factor. In fact, n is usually so big that, by the time the owner has used all possible blinding factors, the underlying cryptosystem is no longer secure.

However, if the owner could use all possible blinding factors, the information available to the agency would be a random permutation of the integers in the interval $[0, n]$. So, even in this extreme case, the agency gains no information about the secret key a. If less than n

different values are used for b, then there is no information gain either.

4 Policies for blinded-key signatures

The blinded-key signature scheme is highly dependent of the policies used to limit key usage. We propose the use of policies based on the following items:

- validity time;
- agency;
- number of signatures issued;
- agreement contents.

In the following subsections, we describe the possible uses for each of these criteria.

4.1 Validity time

A simple policy strategy is to define a short time period where a blinded key can be used. This means that, when the user generates a new blinded key and registers its blinding factor with the notary, the user can state a time period in which the agency is allowed to accept signatures generated with the blinded key. The notary only considers valid those signatures which the agency verifies within this time period. Any signature the agency tries to verify after the validity period is considered as not valid.

To achieve this, the notary must either keep a table of valid signatures or append a signed time-stamp to the signature before returning it to the agency. If the notary is the entity that will process and accept the signature, as in the example of Section 7, the best approach is keeping a table. If the verified signatures are to be forwarded to another party, signed time-stamps may be the best approach.

Validity time based policies can be very effective when used in conjunction with *time-limited black boxes*, which are explained in Section 6.

4.2 Agency restricting policies

Policies for blinded-key signature schemes can also restrict which agencies are allowed to present agreements signed with a certain blinded key. This way, if the owner knows with which agencies the agent will be negotiating, it is possible to restrict the validity of blinded-key signatures output by the agent to those agencies. This precludes another agency from intercepting the agent on the network and making it sign an undesired agreement.

4.3 Number of signatures

Also to limit the possible damage that can be done with a blinded key, the owner may specify how many valid signatures can be generated with a certain blinded key. The notary only accepts as valid the n first signatures it verifies, where n is the maximum number of signatures that can be generated with the given key. This policy prevents an agency from generating more signatures than the user would permit or forwarding the agent to some untrustworthy agency in order to obtain some illicit gain.

4.4 Contents of the agreement

Another limitation on the use of blinded keys can be put on the contents of the agreement, for example, which items can be bought are sold by signing with the given blinded key. This implies that agreements must be built in a standardized format to allow the notary to verify if their contents fits what the policy mandates. Using content limitation for agreements prevents agencies from forcing agents to sign agreements different from what the agent owner did specify in the agent's code.

5 Privacy-preserving blinded-key signatures

Blinded-key signatures may also be used to protect the privacy of the owner against the agency without changing much of the protocol. Two levels of privacy-preserving blinded-key signatures may be provided. In the first level, the owner simply does not include its own identification nor its public key in the agent. All other steps are performed according to the protocol in Section 2, including the signature verification step, in which the agency contacts the notary by sending only the agent's ID, used for fetching the blinding factor from the notary's database. In other words, as the protocol does not allow the agency to verify the signatures by itself, there is no need to send any identification of the owner to the agency. The agent's ID will work as a pseudonym for its owner.

A second level of privacy-preserving blinded-key signatures would allow the owner to generate a new pseudonym for itself and binding it to the true identity. This way, the owner could generate a new key pair and ID for each agent or group of agents. Only the notary would be able to link this pseudonym to the owner's real ID. To increase the level of anonymity, the pseudonym binding notary could be separated from the notary responsible for storing blinding factors.

6 Related Work

This work is related to two different areas of system security research: variants of cryptographic signature schemes, including proxy signatures, and mobile agent security. The first area deals with algorithms and protocols that modify well known digital signature schemes to provide novel uses of the notion of signed messages. The second area, mobile agent security, is concerned with the security implications of widespread adoption of mobile agent systems.

6.1 Variants of cryptographic signature schemes

This area of cryptographic research includes works as different as Chaum's *blind signatures* [Chaum, 1982] or distributed key generation and signing of documents (see [Wu et al., 1999] for an example of distributed signing and [Boneh and Franklin, 1997] for distributed key generation) as well as *proxy signature* schemes [Mambo et al., 1996].

Our blinded-key signatures are a kind of proxy signatures, even if they do not fulfill all the usual requirements for proxy signatures, even if they fulfill the most important requirements. In traditional proxy signature schemes, it may be desired that even the original signer (the entity whose identity is being bound to the information being signed) can not generate fake proxy signatures (signatures generated by the proxy). This makes no sense for blinded-key signatures since the agent owner could execute his own agents and have them sign anything he/she wants.

Blinded-key signatures are also similar to Chaum's blind signatures as both use blinding factors to guarantee the secrecy of information. But whereas blind signatures are meant to conceal the contents of the message being signed from the entity that applied its signature, blinded-key signatures conceal the signing key from the entity which performs the signature computation. For more details on blind signatures and their applications, see [Chaum, 1982].

6.2 Black boxes and mobile agent security

In [Chess, 1998], the author surveys the main security issues in systems with mobile code, including mobile agents. Several aspects of host protection against malicious agents are presented, including authentication, trust and secure languages. Results in protecting the agents against malicious hosts, which is the problem we consider here are also presented in [Chess, 1998].

The main tools for protecting agents against malicious hosts use the concept of black boxes [Hohl, 1998], which guarantees that:

- agent code and data cannot be read (in a meaningful way);
- agent code and data cannot be modified (in a meaningful way).

Two approaches have been proposed to construct black boxes: *time limited black-boxes* [Hohl, 1998] and *mobile cryptography* [Sander and Tschudin, 1998], which includes computing with encrypted functions and computing with encrypted data. Arguing that current cryptographic methods to construct black boxes are neither practical nor general enough, [Hohl, 1998] proposes time limited black-boxes, which are based on code and data obfuscation techniques. In Hohl's approach, black boxes are valid only during a time interval, in which the black box properties remain valid. After the time validity expires, it must be guaranteed that future access to its contents has no unwanted effects. This precludes the inclusion of private keys in agents based on this technique.

Two approaches have been proposed for implementing mobile cryptography. [Sander and Tschudin, 1998] propose a scheme for encrypting mobile agents based on homomorphic cryptosystems, which would allow a remote host to execute agents without being able to infer their contents. [Cachin et al., 2000] construct a black box system combining oblivious transfer and secure function evaluation. Both methods are difficult to understand and to implement and can hardly be considered practical.[1]

[1]In fact, the method of [Sander and Tschudin, 1998] is not practical at all.

6.3 Blinded-key signatures and time-limited black-boxes as complementary solutions

Creating black boxes from mobile agents would solve many of the problems that arise by the execution of agents in an untrusted environment. Unfortunately, current black box approaches have important limitations: their are either not practical or have a limited time span. We focus then on time-limited black-boxing techniques and show how they can be combined with blinded-key signatures to increase the overall security of mobile agents.

The time-limited black-boxes help in protecting the contents of mobile agents during their validity period, based on the difficulty of recovering the original code and data from its obfuscated form. After the expiration of the black box, it cannot be assumed that agent code or data have any kind of protection against disclosure to malicious hosts. Hence, this method still does not allow the inclusion in agents of long-lived private information, such as the user's private key.

Using a combination of time-limited black-boxes and blinded keys, i.e. by first blinding private keys, inserting them in agents and then creating a black box with the agent, it is possible to achieve both protection of the agent code and data and allow the inclusion of private keys in agents. In such a setting, black-boxing prevents the agency from having access to agent negotiating strategy through its code, while blinded-key signatures allow agents to autonomously sign contracts or payment orders on behalf of their owners, achieving a good combination of implementable security and flexibility. This implies that the policy bound to the blinded-key must have at least a validity period component as discussed in Section 4.1. The notary would record and accept as valid only those signatures that it verifies within this validity period.

7 Example use for blinded-key signatures

Mobile agents are being increasingly used for executing electronic commerce transactions and, in order to be really autonomous, they need to be able to execute payment transactions. Using blinded-key signatures, it is possible to adapt existing electronic payment systems for use with mobile agents. In this section, we sketch a simplified account-based electronic payment system, modeled after real-world systems and then show that blinded-key signatures and black-boxing can make it suitable to be used by mobile agents.

7.1 A simplified payment system

Most electronic payment systems are based on the transfer of monetary value between accounts. More precisely, in those systems, the payer gives the payee some electronic token that authorizes the payee to request a fund transfer to the controlling entity for the payment system. Usually those electronic tokens are signed with the payer's private key and before being sent to the payee.

The usual steps for such a payment system are:

1. the payer generates the payment token, which is often called an *electronic check*.

2. the payer sends the payment token to the payee.

3. the payee sends the payment token to the bank, asking it to execute the fund transfer.

7.2 Adapting the simplified payment system to mobile agents

Using blinded key signatures, it is possible to adapt the simplified payment system described in the previous section to a mobile agent environment. To keep the example simple, we assume that both the owner and the agency have asymmetric key pairs which public components are known to the notary. We also assume that the notary and the bank are the same entity.

In our example system agents are protected by time-limited black-boxing techniques and blinded key policies have both validity times, agency restrictions and can only be used once. In addition to those restrictions, the policy also states the maximum amount of payment orders signed by the specific blinded key and the kinds of items being negotiated.

In these new settings, a payment transaction is executed as follows:

1. the owner creates an agent, blinds its private key and releases the agent on the network.

2. the agent negotiates the items and agrees with the content of the payment order. The payment order must clearly state the agency and blinded key identifications, the items being purchased and their prices.

3. the agent signs the payment order and hands it to the agency.

4. the agency verifies the payment order validity with the notary/bank, which will execute the fund transfer if the payment order is valid.

5. the agency releases the agent and assures the items purchased are delivered. The delivery may be guaranteed by the system if the agency is not credited for the purchase until it can prove the items have been delivered.

8 Conclusion

We presented a new cryptographic technique to be used in mobile agent systems called blinded-key signatures. This scheme allows users to embed private keys in autonomous agents in a way that does not compromise the secrecy of their private keys. Used in conjunction with black-boxing techniques, blinded-key signatures allow a good level of protection of agent code and data against malicious hosts. Future work should involve applications of the technique presented here such as agent-based payment systems or contract signing policies.

Autonomous mobile agents will become increasingly important in electronic commerce settings as different as PDA-based shopping and negotiation-based systems, such as auctions or securities trading. However, before mobile agents become a prevalent technology, many problems related to their security must be addressed. Although many practical advances have been made to protect remote hosts against malicious agents, the same is not true for the protection of agents against malicious hosts. Often believed to be unsolvable, this problem is getting more attention lately and new results are beginning to appear. We hope the results described contribute in building what will be the next generation of electronic commerce environments.

References

[Boneh and Franklin, 1997] Boneh, D. and Franklin, M. (1997). Efficient generation of shared RSA keys. In Kaliski, B., editor, *Advances in Cryptology - Proceedings of Crypto'97*, pages 425–439. Springer-Verlag.

[Cachin et al., 2000] Cachin, C., Camenisch, J., Kilian, J., and Müller, J. (2000). One-round secure computation and secure autonomous mobile agents. In *Proceedings of ICALP'2000*.

[Chaum, 1982] Chaum, D. (1982). Blind signatures for untraceable payments. In Chaum, D., Rivest, R. L., and Sherman, A. T., editors, *Advances in Cryptology: Proceedings of Crypto'82*. Springer-Verlag.

[Chess, 1998] Chess, D. M. (1998). Security issues in mobile code systems. In Vigna, G., editor, *Mobile Agents and Security*, number 1419 in Lecture Notes in Computer Science, pages 1–14. Springer-Verlag.

[Hohl, 1998] Hohl, F. (1998). Time limited blackbox security: Protecting mobile agents from malicious hosts. In Vigna, G., editor, *Mobile Agent Security*, number 1419 in Lecture Notes in Computer Science. Springer-Verlag.

[Mambo et al., 1996] Mambo, M., Usuda, K., and Okamoto, E. (1996). Proxy signatures: Delegation of the power to sign messages. *IEICE Transactions Fundamentals*, E79-A(9):1338–1354.

[Menezes et al., 1997] Menezes, A. J., Oorschot, P. C., and Vanstone, S. A. (1997). *Handbook of Applied Cryptography*. CRC Press. Available online at http://www.cacr.uwaterloo.ca/hac.

[Rivest et al., 1978] Rivest, R. L., Shamir, A., and Adleman, L. M. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[Sander and Tschudin, 1998] Sander, T. and Tschudin, C. F. (1998). Protecting mobile agents against malicious hosts. In Vigna, G., editor, *Mobile Agent Security*, number 1419 in Lecture Notes in Computer Science. Springer-Verlag.

[Wu et al., 1999] Wu, T., Malkin, M., and Boneh, D. (1999). Building intrusion tolerant applications. In *Proceedings of the 8th USENIX Security Symposium*, pages 79–91.