Blinding for Unanticipated Signatures

David Chaum

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam

Summary

Previously known blind signature systems require an amount of computation at least proportional to the number of signature types, and also that the number of such types be fixed in advance. These requirements are not practical in some applications. Here, a new blind signature technique is introduced that allows an unlimited number of signature types with only a (modest) constant amount of computation.

Background

The notion of "blind signatures" was first introduced in [Chaum 82]. Such systems typically have a party called the *signer* who is able to make certain digital signatures. The other parties, called *providers*, would like to obtain such signatures on messages they provide to the signer. What makes it interesting is that the providers wish to keep the signer in the dark as much as possible. Since the signatures will in general ultimately be shown widely, it is assumed that the signer will learn them. What providers can keep from the signer, however, is the exact correspondence between the actual signing operations performed by the signer and the signatures which are later made public. This essential property of blind signatures is called "unlinkability."

Blind signatures can be made using the RSA public key signature system [Rivest, Shamir & Adleman 78]. As is usual for such systems, the *signer* chooses two appropriate large primes p and q, and makes their product n public. In a slight generalization, which will be important later, the signer then issues l "public exponents" $e_1, ..., e_l$. The signer further computes corresponding "secret exponents" $d_1, ..., d_l$ satisfying $d_i \equiv e_i^{-1}$ (mod $(p-1)(q-1)$), $1 \leqslant i \leqslant l$. The signature of "type" i is formed on a number m by the signer as $m'_i \equiv m^{d_i}$ (mod n). Anyone can use the public n and e_i to verify the signature of type i on m by checking that $m \equiv (m'_i)^{e_i}$ (mod n) holds.

The first actual realization of a blind signature system [Chaum 85] worked as follows: The provider "blinds" a message m, using a key k chosen uniformly from $\{1,..,n\}$, producing $t \equiv [m]k^{e_i}$ (mod n), where the square brackets indicate the input to the transformation. The signing of t yields $t' \equiv [mk^{e_i}]^{d_i} \equiv m^{d_i}k \pmod n$. Then the provider "unblinds" this returned t' by forming $m'_i \equiv [m^{d_i}k]k^{-1} \equiv m^{d_i} \pmod n$. It is easy to see informally why the unlinkability property mentioned above holds. The signer knows two sets: the set of signed blinded messages t' (or equivalently to the signer, t) and the set of signed messages m'_i (or equivalently, m_i). For any particular t' and m'_i that the signer suspects might correspond, the signer is able to determine exactly one $k \equiv t'm'^{-1}_i$ (mod n) that would have been the provider's secret blinding key if they did in fact correspond. But since the k's are chosen independently and uniformly by the providers, the signer learns nothing about whether any guessed correspondence holds.

Notice that a provider must "anticipate" the particular d_i that will be used by the signer. It is possible, though computationally expensive, for the provider to actually anticipate a few d_i at once by forming $t \equiv mr^{e_1e_2} \pmod n$ for example, and being able to unblind in case of signature with d_1 or d_2 by forming $m'_1 \equiv (mr^{e_1e_2})^{d_1}\ r^{-e_2} \pmod n$ or $m'_2 \equiv (mr^{e_1e_2})^{d_2}\ r^{-e_1} \pmod n$, depending on whether d_1 or d_2 was used to sign, respectively. But such an approach becomes prohibitively computation intensive as the number of alternatives increases: in general it requires the provider to perform more than one multiplication for each alternative anticipated (since each e_i should have a distinct prime factor, otherwise some signatures can be made from others). Such effort required to anticipate all possible signatures may not be practical, and is also undesirable because the maximal extent of a system has to be fixed initially and effort proportional to this limiting size must be carried out during every blinding. Such an approach of course becomes impossible in practice when the number of alternatives is too large or when the alternatives are not known in advance of blinding.

The ability to anticipate a large number of signature types can benefit the payment system described in [Chaum 85]. This would allow customers of the bank providing a system to each supply a large number of blinded items when their accounts are opened, without the customers knowing in advance which particular type of signature will later be applied by the bank. Not only can this provide economy of data transfer, but it protects the bank's customers from being able to (and hence from being coerced into) making payments that they cannot later trace. When the bank ultimately issues signed notes, its choice of signature type may be used, for example, to encode denominations and expiration dates. Another application of blind signatures is to credential systems. In these, each different kind of credential a person receives is encoded as a digital signature of a corresponding type formed on a blinded copy of the person's "digital pseudonym" used with the signer [Chaum 85; Chaum & Evertse 86]. There may be a great many different kinds of basic and more detailed credentials, and the future requirement for such credentials could be difficult to anticipate. Thus there appears to be a substantial need for blind signatures that are unblindable even after an unanticipated signature type has been applied.

Blind unanticipated signature protocols

The new blind signature protocols presented here are based on the use of certain units (of the residue classes modulo N) called *generators*. These will be made public in advance, and their signatures will also be made public by the signer. Unlinkability requires that the group they generate contains all messages m (otherwise certain linkings are easily ruled out because there is no k capable of causing them). Depending on how this requirement is ensured, as will be described later, the blind unanticipated signature protocol will have one of three forms.

In the first and simplest form of these protocols, the following congruences hold:

$$t \equiv [m] g^k \pmod n$$

$$t' \equiv [mg^k]^{d_i} \pmod n$$

$$m' \equiv [(mg^k)^{d_i}] g^{-d_i k} \pmod n,$$

where: m is the message to be blinded; t is the blinded form of the message; t' the signature computed for t; m' the unblinded signed m; g the generator; n the publicly known modulus; e_i and d_i the public and private signature exponents, respectively; and k the provider's secret blinding key, say chosen uniformly from $\{1, \cdots, n^2\}$. The square brackets again show the input to the transformation whose output is shown on the left-hand-side, and thus they define the function of each of the three transformations in the order shown: blinding, signing, and unblinding.

In the second form of the protocol, the following congruences hold:

$$t \equiv [m] g^{k_1} k_2 \pmod n$$

$$t' \equiv [mg^{k_1} k_2]^{d_i} \pmod n$$

$$m' \equiv [(mg^{k_1} k_2)^{d_i}] g^{-d_i k_1} k_2 \pmod n,$$

where blinding key k_1 is chosen as k was above, but blinding key k_2 is chosen uniformly only from the pair $\{1, n-1\}$.

In the third and most general case, the following congruences hold:

$$t \equiv [m] g_1^{k_1} g_2^{k_2} \ldots g_r^{k_r} \pmod{n}$$

$$t' \equiv [m g_1^{k_1} g_2^{k_2} \ldots g_r^{k_r}]^{d_i} \pmod{n}$$

$$m' \equiv [(m g_1^{k_1} g_2^{k_2} \ldots g_r^{k_r})^{d_i}] g_1^{-d_i k_1} \ldots g_r^{-d_i k_r} \pmod{n},$$

where each k_i is a secret blinding key chosen independently as k was above.

Testing a generator knowing the factorization

If generators for the whole group are known, then unlinkability can be maintained without any restriction on the messages that can be signed. Having the factorization of $p-1$ and $q-1$ allows efficient verification that a particular proposed set of generators do in fact generate the whole group of units modulo n. (Of course these factorizations also allow p and q to be computed easily.) Suppose, for example, that GCD($p-1$, $q-1$)=2 and $q \equiv 3 \pmod 4$ (which is quickly checked by comparing the factors of $p-1$ and $q-1$) and that g and -1 are to be verified as generating the whole group (so that the second protocol above can be used). Then it is sufficient to verify that the order of g is maximal modulo one factor of n, say p, and that it generates exactly half the elements modulo the other. This is readily accomplished: first raise g to $p_l^{-1}(p-1)$ and ensure that no l, $1 \leqslant l \leqslant k$, gives the result 1, where the p_i, $1 \leqslant i \leqslant k$, are the prime divisors of $p-1$; perform the same procedure also for the odd primes dividing q.

Hiding the factorization with physical security

The above approach works fine if you know the factors, but how can it be used by the signer to convince providers without revealing the factorization? A possible solution allows anyone to submit to the signer a physical device for conducting the above test. The signer would give the secret parameters to the device and allow the message containing the logical result ("verifies" or "does not verify") to be sent by the device to its supplier. This should be done in a way that convinces the supplier that the signer cannot falsify the message, while not allowing the device to leak the secret parameters. One way to arrange this is as follows: The supplier creates a number c' at random, applies a publically known and agreed on one-way function f, which is preferably one-to-one, to c', yielding $c = f(c')$; installs c' and n in the device; and gives c to the signer along with the device. Then the signer isolates the device from the supplier and feeds it the secret parameters (p and q). Only if the result is "verifies" does the device output c'. The signer then applies f to the output of the device, and checks whether the result is c, and if so returns this c' to the supplier. The device is constructed to be tamper resistant enough to make it sufficiently difficult for the signer to obtain c' from it in the expected time interval, unless the generator tester yields "verifies."

Linear cut-and-choose on moduli and generators

In another approach, a prospective signer announces moduli n_i, $1 \leqslant i \leqslant k$, and corresponding generators g_i for each (the "cut"). Others determine the single modulus n among these that will be used (the "choose"), whereupon the signer must make public the factorization of the $p_l - 1$ and the $q_l - 1$, $1 \leqslant l \leqslant j$ and $l \neq i$, for all j moduli except the ith one selected. Then anyone can use the verification techniques described above (which rely on possession of the secret) to verify all the other moduli and corresponding generators, thereby obtaining the probability $1 - j^{-1}$ that the selected modulus would also verify. Such techniques can of course also be used in a great many situations, as is well known in the folklore. One way to select i would be by a public coin-tossing event. Another way to determine i requires each of some set of persons to form a b' at random and a corresponding $b = f(b')$ using a preferably bijective, public one-way function f. Then each participant makes public their b. Once every b is public, the b' are revealed, checked, and added modulo the number of moduli j, yielding the index i. A disadvantage of these techniques, compared to the previous approach based on physical devices, is that they are not "open" to new participants once the cut-and-choose is completed.

Generators not manipulable by the signer

Yet another way for providers to gain confidence in the suitability of generators is for the generators to be determined in a way that cannot easily be manipulated by anyone including the signer. Generators could be chosen by a random process, such as those just described for selecting a particular modulus. Use of a single such randomly chosen and untested generator might not give a high enough probability of providing adequate unlinkability; use of several randomly chosen generators improves the degree of hiding and unlinkability. For instance, it can be shown that use of 22 generators provides a probability of roughly one-millionth that not all blinding factors in the reduced residue system modulo n are generated. When additional things are known about the structure of n, the probabilities improve greatly and less generators need be used. Examples include: when n has exactly two prime factors each congruent to 3 modulo 4 (which can be demonstrated efficiently by a protocol presented by [Peralta & v.d. Graaf 87]) and / or when $(p-1)(q-1)$ has no small odd divisors (as is easily checked by asking the signer to sign random values using signature types with small prime e_i). A further variation chooses subsets of a set of generators in a key-dependent way.

Ensuring that messages blinded are powers of g

This final approach does not require verification or special selection of the generator(s): protection against linking is provided by almost any generator(s). Security derives from testing whether the thing to be signed, m, is a member of the group generated by the generator(s). This can be used when the signer itself forms m as $m \equiv g^x$ (mod n), where x is chosen at random by the signer (such as is the case in the setting of [Chaum & Evertse 86]). Then the signer can participate in a protocol to convince the provider that some x satisfying the congruence is known to the signer. The first protocols for demonstrating such "possession of a discrete log" are presented in [Chaum, Evertse, v.d. Graaf & Peralta 86], and a potentially more efficient version allowing many m's to be verified at once is presented in [Chaum, Evertse & v.d. Graaf 87].

Unlinkability

It seems that all blinding factors cannot be equally likely, since otherwise the provider would have to choose the exponents from a distribution depending exactly on LCM($p-1$, $q-1$). But it is easy to see that choosing the k's uniformly from $\{1, \cdots, n^2\}$ for instance gives a maximum difference in the probability of any two that is less than n^{-2}, which would be quite adequate in practice.

Open problems

A much cleaner and more general approach might be found. Notice, for example, that if $(p-1)/2$ and $(q-1)/2$ are primes, then any g with Jacobi symbol -1 (which can easily be checked knowing only n) together with -1 generates the whole group (apart from a few exceptional cases that immediately reveal the factorization of n). Thus, a protocol allowing anyone interacting with the signer to check that n is of such a form without learning its factorization allows generators of the whole group to be verified directly. Any such open protocol for establishing a pair of generators (preferably one of which is -1) would of course be quite desirable.

Acknowledgements

I am indebted to Jan-Hendrik Evertse for substantial collaboration in related areas during the time that the basic results presented here were developed. Subsequently, Jan-Hendrik Evertse, Jeroen van de Graaf, and René Peralta worked with the author to find improved techniques. Final revisions also benefited from the comments of Jurjen Bos, Bert den Boer and Gilles Brassard.

References

(1) Chaum, D., "Blind signatures for untraceable payments," *Proceedings of Crypto 82,* D. Chaum, A. Sherman & R. Rivest, Eds., Plenum 1983, pp. 199-203.

(2) Chaum, D., "Security without identification: transaction systems to make big brother obsolete," *Communications of the ACM,* 28, 10 (October 1985), pp. 1030-1044.

(3) Chaum, D. & Evertse, J.-H., "A secure and privacy-protecting protocol for transmitting personal information between organizations," *Proceedings of Crypto 86,* A.M Odlyzko Ed., Springer-Verlag 1987, pp. 118-167.

(4) Chaum, D., Evertse, J.-H., van de Graaf, J., & Peralta, R., "Demonstrating possession of a discrete logarithm without revealing it," *Proceedings of Crypto 86,* A.M Odlyzko Ed., Springer-Verlag 1987, pp. 200-212.

(5) Chaum, D., Evertse, J.-H. & van de Graaf, J., "An improved protocol for demonstrating possession of a discrete logarithm and some generalizations," *Proceedings of Eurocrypt 87.*

(6) van de Graaf, J. & Peralta, R., "A simple and secure way to show the validity of your public key," to appear in *Proceedings of Crypto 87.*

(7) Rivest, R., Shamir, A. & Adleman, L. "A method for obtaining digital signatures and public-key cryptosystems" *Communications of the ACM,* 21, 2, (February 1978), pp. 120-126.