Online Cash Checks

David Chaum

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam

INTRODUCTION

Savings of roughly an order of magnitude in space, storage, and bandwidth over previously published online electronic cash protocols are achieved by the techniques introduced here. In addition, these techniques can increase convenience, make more efficient use of funds, and improve privacy.

“Offline” electronic money [CFN 88] is suitable for low value transactions where “accountability after the fact” is sufficient to deter abuse; online payment [C 89], however, remains necessary for transactions that require “prior restraint” against persons spending beyond their available funds.

Three online schemes are presented here. Each relies on the same techniques for encoding denominations in signatures and for “devaluing” signatures to the exact amount chosen at the time of payment. They differ in how the unspent value is returned to the payer. In the first, all change is accumulated by the payer in a single “cookie jar,” which might be deposited at the bank during the next withdrawal transaction. The second and third schemes allow change to be distributed among unspent notes, which can themselves later be spent. The second scheme reveals to the shop and bank the maximum amount for which a note can be spent; the third does not disclose this information.

DENOMINATIONS AND DEVALUING

For simplicity and concreteness, but without loss of generality, a particular denomination scheme will be used here. It assigns the value of 1 cent to public exponent 3 in an RSA system, the value of 2 cents to exponent 5, 4 cents to exponent 7, and so on; each successive power-of-two value is represented by the corresponding odd prime public exponent, all with the same modulus. Much as in [C 89], a third root of an image under the one-way function f (together with the pre-image modulo the bank’s RSA composite) is worth 1 cent, a 7th root is worth 4 cents, and a 21st root

5 cents. In other words, a distinct public prime exponent is associated with each digit of the binary integer representation of an amount of payment; for a particular amount of payment, the product of all those prime exponents corresponding to 1's in the binary representation of the amount is the public exponent of the signature.

A signature on an image under f is "devalued" by raising it to the public powers corresponding to the coin values that should be removed. For instance, a note having a 21st root could be devalued from its 5 cent value, to 1 cent, simply by raising it to the 7th power.

In earlier online payment systems [C 89], the number of separate signatures needed for a payment was in general the Hamming weight of the binary representation of the amount. Since online systems would be used for higher-value payments (as mentioned above), and extra resolution may be desired to provide interest for unspent funds [C 89], an average of roughly an order of magnitude is saved here.

COOKIE JAR

In this first scheme the payer periodically withdraws a supply of notes from the bank, each with the system-wide maximum value. Consider an example, shown in Figure 1.1, in which two notes are withdrawn. The n_i and r_i are random. The r_i "blind" (from the bank) the images under the public, one-way function f . The bank's signature corresponds to taking the h th root, where $h = 3 \cdot 5 \cdot 7 \cdot 11$. As in all the figures, the payer sends messages from the left and the bank sends from the right.

Fig. 1.1. Cookie-jar withdrawal

In preparing the first payment, the payer divides r_1 out. The signature is then raised to the 55th power to devalue it from 15 cents to 5 cents. Figure 1.2 shows this first payment. Of course the shop is an intermediary between the payer (left) and the bank (right) in every online payment, but this is not indicated explicitly. Also not shown in the figures are messages used to agree on the amounts of payment.

Fig. 1.2. First cookie-jar payment

The first two residues sent in paying, n_1 and its signed image under f , are easily verified by the bank to be worth 5 cents. The third residue is a blinded "cookie jar," a blinded image under f of a randomly chosen value j . This cookie jar is modulo a second RSA composite that is only used for cookie jars. Once the bank verifies the funds received, and that n_1 has not been spent previously, it signs and returns the blinded cookie jar (under the cookie-jar modulus) with public exponents corresponding to the change due.

The second payment, shown in figure 1.3, is essentially the same as the first, except that the amount is 3 cents and the cookie jar now has some roots already on it. If more payments were to be made using the same cookie jar, all resulting signatures for change would accumulate.

Fig. 1.3. Second cookie-jar payment

The cookie jar might conveniently be deposited, as shown in figure 1.4, during the withdrawal of the next batch of notes. It is verified by the bank much as a payment note would be: the roots must be present in the claimed multiplicity and the pre-image under f must not have been deposited before.

Fig. 1.4. Cookie-jar deposit

The cookie-jar approach gives the effect of an online form of “offline checks” [C 89], in that notes of a fixed value are withdrawn and the unspent parts later credited to the payer during a refund transaction.

DECLARED NOTE VALUE

Figure 2 depicts a somewhat different scheme, which allows change to be spent without an intervening withdrawal transaction. Withdrawals can be just as in the cookie-jar scheme, but here a single modulus is used for everything in the system. The products of public exponents representing the various amounts are as follows: d is the amount paid, g is the note value, the “change” c is g/d , and h is again the maximal amount, where $d \mid g \mid h$. A payment (still to the bank through a shop) includes first and second components that are the same as in the cookie-jar scheme. The third component is the amount of change c the payer claims should be returned. The fourth is a (blinded) number m , which could be an image under f used in a later payment just as n is used in this one.

Fig. 2. Declared note value payment

The signature returned contains a “protection” factor (shown inside the padlock). This factor ensures that the payer actually has the c^{th} root of $f(n)$, by requiring that the payer apply f to it before dividing the result out of the signature. Without such protection, a payer could get the system-wide maximum change, regardless of how much change is actually due; with it, the change claimed can only be recovered if the corresponding roots on n are in fact known to the payer.

DISTRIBUTING CHANGE

The change returned in a payment can be divided into parts that fill in missing denominations in notes not yet spent. Suppose, for example, that the last payment is spent with $d = 5.11$, $c = 3.7$, and that m is formed by the payer as shown in the first line of Figure 3.1. Then unblinding after the payment yields the a shown in the second line.

$$m \equiv f(n_1)^3 f(n_2)^7$$

$$a \equiv m^{1/21}$$

Fig. 3.1. Form of change returned

From a , the two roots shown in the last two lines of Figure 3.2 are readily computed. (This technique is easily extended to include any number of separate roots.) Thus the values unused in the last payment fill in roots missing in notes n_1 and n_2 .

$$u = 3^{-1} \bmod 7$$

$$v = 3u \bmod 7$$

$$f(n_1)^{1/7} \equiv (a^3 f(n_2)^{-1})^u f(n_1)^{-v}$$

$$f(n_2)^{1/3} \equiv a f(n_1)^{-1/7}$$

Fig. 3.2. Distributing the change

Because overpayment allows change to be returned in any chosen denominations (not shown), the payer has extra flexibility and is able to use all funds held. This also increases convenience by reducing the need for withdrawals.

HIDDEN NOTE VALUE

Although the combination of the previous two subsections is quite workable, it may be desirable for the payer not to have to reveal c to the shop or the bank. Figure 4 shows a system allowing this. The payment message is just as in the declared note value protocol above, except that c is not sent. The protection factor (shown again in a lock) is also placed under the signature, but it is missing the extra f and is raised to a random power z chosen by the bank.

Fig. 4. Hidden note value payment

If z were known to the payer before payment, then the payer could cheat by including $f(n)^{-z}$ in the third component; this would yield the payer the system-wide maximum change, even if none were due. Consider a single change exponent q . If $z \bmod q$ is guessed correctly by a cheating payer, then the payer improperly gets the corresponding coin value. Thus, the chance of successful cheating is $1/q$. If, however, the divisors of h are chosen sufficiently large, quite practical security can be achieved. When the possibilities of distributing change and refunding are included, this scheme's privacy surpasses that of a coin system.

CONCLUSION

Combining online coins improves efficiency, use of funds, convenience, and privacy.

REFERENCES

- Chaum, D., "Privacy Protected Payments: Unconditional Payer and/or Payee Anonymity," in *Smart Card 2000*, North-Holland, 1989, pp. 69–92.
- Chaum, D., A. Fiat, & M. Naor, "Offline Electronic Cash," *Proceedings of Crypto '88*.