Extensions to Chaum's Blind Signature Scheme and OpenCoin Requirements

A.W. Dent, K.G. Paterson and P.R. Wild,
Information Security Group,
Royal Holloway, University of London

27th February 2008

1 Introduction

Section 2 of this document gives an overview of the Chaum on-line e-cash scheme that makes use of blind RSA signatures. Section 3 outlines some extensions of the basic Chaum scheme that could be of use in the OpenCoin project. Section 4 discusses future requirements of the OpenCoin project and how they might be met.

2 The Chaum E-cash Scheme

In this section we review the Chaum [6] concept of using blind RSA signatures for untraceable payments and e-cash.

2.1 The Basic Scheme

Chaum's scheme uses RSA signatures. There are three participants: the *Issuer* IS who issues e-coins to a *User* U who wants to use them to pay a *Merchant* M. An important attribute of the scheme is *user anonymity*: the use of an e-coin should not provide any information about the identity of the User. The technique involves the User preparing a *Blank* that is blinded before being sent to the Issuer to sign and return to the User. The User removes the blinding to obtain a signed Blank, *i.e.* a *Token*. The User gives the Merchant the Token in exchange for goods and the Merchant returns the Token to the Issuer to redeem the value once the Issuer has verified that the signature is valid.

The Issuer generates an RSA modulus n, the product of two (large) primes p and q, and a pair of public and private keys, e and d, such that $ed = 1 (\mathrm{mod}\ \phi(n))$. The pair n, e is made public in an authenticated way and d is kept secret.

The set of Blanks is a subset $B \subseteq \mathbf{Z}_n^*$. The set of Tokens is

$$C = \{m^d (\mathrm{mod}\ n) | m \in B\}.$$

The User chooses an element of B at random and a blinding element $r \in \mathbf{Z}_n^*$ at random and calculates the blinded Blank $c_1 = r^e m (\mathrm{mod}\ n)$. The User and Issuer engage in an exchange in which the User pays the Issuer and the Issuer mints a blinded Token $c_2 = c_1^d (\mathrm{mod}\ n)$ to give to the User. How this exchange takes place and whether or not the User and Issuer trust each other is left for later discussion. Nevertheless the User can verify that the Issuer has used the private key d by checking that $c_2^e = c_1 (\mathrm{mod}\ n)$. Payment may be made using 'real cash', by a debit from a bank account of the User held with the Issuer, or possibly by a Token (e-coin). The User unblinds the blinded Token to obtain the Token $t = m^d = c_2 r^{-1} (\mathrm{mod}\ n)$.

When a User spends an e-coin at a Merchant, the User gives the Merchant a Token t in exchange for goods. How this exchange takes place and whether or not the User and Merchant trust each other is left for later discussion. Nevertheless the Merchant can verify that the Token has been signed by the Issuer. The Merchant sends the token t to the Issuer in order to redeem its value. The Issuer maintains a list of Blanks that have been spent. The Issuer verifies that the signature is valid by checking that $m = t^e \in B$ and that m has not been spent. If this is so then the Issuer pays the Merchant the value of the token (this may be by credit to a bank account or by minting an e-coin for the Merchant or by some other means – how the exchange between the Merchant and the Issuer takes place and whether or not the Merchant and Issuer trust each other is left for later discussion). This process may or may not take place before the exchange between the User and the Merchant completes.

It is clear that if the User, the Merchant and the Issuer all act honestly then in the execution of the scheme as described the User does pay the Merchant for goods received.

2.2 Unforgeability

It is essential that only the Issuer, using the private key d, should be able to construct a Token $t = m^d (\mathrm{mod}\ n)$ for a Blank $m \in B$ other than deriving it from a blinded token obtained from the Issuer. This should be so even with the knowledge of other Blank, Token pairs (for chosen Blanks or Tokens). This places conditions on the set B. For example, it should not be possible to construct Blanks $m_1, m_2 \in B$ such that $m = m_1 m_2 (\mathrm{mod}\ n) \in B$ since then by obtaining Tokens t_1, t_2 corresponding to m_1, m_2 from the Issuer, the User may construct a token $t = t_1 t_2 (\mathrm{mod}\ n)$ corresponding to m, *i.e.* succeed in the one-more-forgery attack.

2.3 Blank Generation

A Blank must both have some redundancy, so that the signature of the Issuer may be verified, and have some randomness, so that the probability that two Users generate identical e-coins is small. This may be achieved with a Blank that is the concatenation of a fixed part f specified by the Issuer and a random string a generated by the User. Thus a Blank has the form $m = f||a$ or $m =$

$h(f||a)$ for some hash function. The User should use a random number generator whose outputs are distinct and unpredictable to produce the strings a. Moreover distinct Users should produce distinct random strings. If a hash function is used then it needs to be collision-resistant.

3 Extensions of Chaum's Scheme

3.1 Randomized Signatures

Since the Issuer signs a blinded Blank c_1 the Issuer cannot be certain that the corresponding Blank has any particular form. An attacker who obtains signatures on spoof blinded Blank values $x_1, \ldots, x_l$ such that many of $x_i x_j \in B$ for $1 \leq i \leq l$, $1 \leq j \leq l$ may potentially succeed in a "one-more-forgery attack". However, we note that the basic scheme of Chaum has been proven to be invulnerable to such an attack in any meaningful way [2].

In a randomized signature the Issuer introduces some random value into the signing process so that the User does not determine (or even know) what value has been signed. This is intended to provide a defence against the one-more-forgery attack. The mechanism is as follows. The Issuer chooses a random value x and sends the User the value $y = x^e (\text{mod } n)$. The User chooses a random value u and constructs the Blank $m = h(f||a||b)$ where $b = u^e y (\text{mod } n)$ and the blinded Blank $c_1 = r^e um (\text{mod } n)$ which is sent to the Issuer. The Issuer returns a pair of values $c_2 = ((c_1 x)^d (\text{mod } n), x)$ as the signed blinded Blank and the User forms the pair $t = (c_3 = c_2 r^{-1} (\text{mod } n), v = ux (\text{mod } n))$ as the Token. A token $t = (c_3, v)$ is verified by checking that $c_3^e v^{-1} = h(f||a||v^e) (\text{mod } n)$.

3.2 Time Limits

As e-coins are spent the Issuer's database of spent Blanks will become unwieldy. Placing a time limit on the validity of a Token means that spent Blanks need not be kept in the database after the expiration of the corresponding Token. An expiry date in the fixed part of a Blank and in the certificate of a denomination's signature verification key may be used for this purpose.

3.3 Double Spending and Online Issuer

An issue with the basic scheme is that the User may behave dishonestly and try to spend a Token more than once. The Issuer will discover this when recording the corresponding Blank for a second time in its database of spent e-coins but the Merchant on its own cannot check that a Token has not been spent before. Thus the Issuer needs to be online to provide the Merchant with an assurance, before the Merchant completes the transaction with the User, that the e-coin he has been offered has not already been spent. On the other hand, since the Merchant could take the Token given by the User as payment and spend it elsewhere, the User may not want to provide the Token and wait, without completing the transaction, while the Merchant obtains this assurance.

In a modification of the Basic Scheme the User gives the Merchant the Blank rather than the corresponding Token when offering payment. The Merchant contacts the online Issuer to check whether the Blank appears on the database of spent e-coins. If it does not then the Issuer places a lock on the Blank so that only the enquiring Merchant may redeem the e-coin by presenting the corresponding Token after completing the transaction with the User. When the Merchant provides the goods in exchange for the Token a verification of the signature will check that it corresponds to the Blank that has been locked. Such a lock should have a time limit (or other mechanism) to enable the User to redeem the Token should the Merchant not complete the transaction.

3.4 Tracing Double Spending

An alternative to having the Issuer online to prevent double spending is to have a means of tracing Users who double spend and to take action against them. This requires that the User's identity is encoded in the Blank in such a way that is only revealed if the User double spends. Chaum *et al.* [7] were the first to describe a scheme that achieves this. Their scheme uses the cut-and-chose method. Here, the User generates many blinded Candidate Blanks and is challenged by the Issuer to reveal information about half of them. A correct response provides the Issuer with an assurance that the User's identity has been encoded in the Candidate Blanks and the Issuer signs the other half of them collectively. When the User gives the Merchant the token the Merchant also challenges the User to reveal information about the Candidate Blanks. A correct response provides an assurance to the Merchant that the Token is valid. Should the User double spend a Token then, with high probability, all the responses given to the Merchants provide the Issuer with the means to identify the User.

Unfortunately the scheme of [7] is complex and computationally intensive. Moreover there are issues relating to the need to trust the Issuer in policing double spending. However, it points the way forward to more modern schemes that offer off-line double spending detection at lower cost. All of these schemes require users in the system to have their own public keys, and hence for these keys to be authenticated using some form of PKI.

3.5 Partially Blind Signatures

An alternative to the use of the key management technique of Section 3.2 to enable the signing of different denominations and the imposing of time limits is the use of partially blind signatures. In a partially blinded signature scheme the User and Issuer agree on some common information a (such as the denomination or an expiry date) which becomes part of the signature and is a required input to a verification of the signature. Abe and Fujisaki [1] have proposed such a scheme. Chien *et al.* [10] have proposed a less computationally intensive scheme but it is vulnerable to the one-more-forgery attack. Cao *et al.* [5] propose another low complexity partially blind signature scheme. Their technique is to use a to mask the signing key d. Thus they have distinct but related signing keys for distinct

denominations (or other common information a). In particlar, given a master key d, they propose signing keys of the form $\tau(a)d$ for some coding function τ with input the common information a. The common information a becomes part of the signature.

4 OpenCoin Requirements

The document `https://trac.opencoin.org/trac/opencoin/browser/trunk/standards/requirements.txt` lists the following as the OpenCoin project's current and future requirements:

- The system shall be resistant to compromising, i.e.
  - Tokens cannot be falsified without having the minting keys
  - 'Anonymity': issuer should not be able to correlate a minted blind to a redeemed token.
  - Everybody should be able to verify if a token is valid (signed by the issuer's mint, fulfills the token format specs of the issuer, not expired, not spent yet, ...)
  - no double spending possible. There shall be not race condition or other trick to circumvent the double spending check.
- Strength against denial of service
  - How to distribute the DSDB (double spending database)
  - How to prevent a malicious receiver from locking a token forever making it unredeemable for its owner?
  - How to minimize impact of temporarily unavailable issuer services (coin expiration!)
- Future directions
  - Are there protocols/algorithms besides Chaum which are more suited?
  - Receipts: How could mutual receipts ("i hereby certify that I sent/received 10 opencents from Alice/to Bob") come into play? Can the protocol be designed in a way such receipts are mandatory? I.e. the issuer can detect if receipts where exchanged and refuses redemption otherwise.
  - Offline tokens: There are ways to debunk double spenders. How do they work? Is it feasible to implement them?
  - Transferable tokens (what's the correct term?): with Chaum's protocol, every receiver should redeem (or exchange) tokens immediately. How to make tokens 'multi-hop' capable?
  - Conditional anonymity: The issuer can detect if blinded tokens are blinded such that a trusted third party (e.g. law enforcement agencies) could unblind them and refuses minting otherwise.

In the next subsections, we comment on each of these requirements in turn.

4.1 The system shall be resistant to compromising

Issues under this topic are standard for e-cash schemes and are certainly attainable by Chaum's on-line scheme as implemented in OpenCoin. The detection of double spending is dependent on maintaining an on-line double spending database (DSDB), and this raises concerns about how quickly merchants are able to deposit coins, compared to how quickly users can try to double spend them. (Note that there is no double-spending detection in the basic Chaum scheme – the coins are truly anonymous, and cannot be opened if double spent, unlike more advanced, off-line schemes.) The OpenCoin team is aware of this issue and has been exploring the available trade-offs between detecting cheating spenders and detecting cheating merchants. At the root of this is the fair-exchange problem.

4.2 Strength against denial of service

Issues here are largely to do with availability and resistance to Denial of Service (DoS) attacks.

The DSDB can be distributed using (presumably) standard techniques in distributed/high availability database design. Having DoS-resistance for the coin redemption and double spending detection steps is crucial, since an attacker who can mount a DoS against these services may be able to spend coins multiple times during the course of a DoS attack without being detected.

It is also worth examining the extent to which the specific cryptographic components are vulnerable to DoS, based, for example, on exhaustion of computational resources. To take one example, the coin issuing protocol requires the mint to carry out a non-trivial cryptographic computation (signing of a blinded coin). However, in the basic protocol, the client requesting a coin need not carry out any computations at all in order to create seemingly valid coin issuing requests: since the submissions are blinded, he can simply submit random strings of the correct size to the mint. This kind of attack can be defended against via an economic disincentive if every run of the minting protocol results in an amount to be debited from the requesting user's account.

The mint (Issuer) is required to sign all e-coins that are minted (and to verify all e-coins when they are spent). This may place a serious burden on the mint even under normal operating conditions. Moreover, a sustained DoS against the minting interface to the mint would prevent new coins being issued. Since coins can only be spent once, the system would eventually grind to a halt once all issued coins were spent. One counter-measure to this threat is to set up multiple servers capable of minting. But the mint's private key is involved in every coin issuing process, and it is a security-critical component, since its compromise would allow an attacker to create arbitrary value in the system without users' accounts being debited. It is therefore advisable to limit the distribution of the mint's private key as much as possible. This is in contradiction to the desire to distribute the coin minting capability.

It may therefore be advisable to use threshold cryptographic techniques in

this situation. Here, the private key of the mint is split into pieces and distributed to sub-mints. This can be done in such a way that any threshold k of sub-mints can, working together, create a (blinded) signature that, when unblinded, verifies under the mint's usual public verification key. However, no group of size smaller than the threshold k can create signatures, thus protecting against any number of sub-mints being compromised up to that threshold value. The blind RSA scheme of Chaum supports this kind of threshold signing in a fairly natural way; details follow.

In a *k-out-of-l Threshold Signature* scheme the ability to create signatures under a private key d is distributed to a group of l signers such that any k of them can produce a signature in cooperation but no fewer than k can do so. In such a scheme the signing key d determines l *shares* $d_1, \ldots, d_l$ and these are distributed in secret to the l signers $P_1, \ldots, P_l$ respectively. These shares will have the property that the signatures on a message m under any k of them may be combined to provide the signature on the message m under the key d.

Shoup [14] presents a threshold signature scheme for RSA in the case where $n = pq$ is a product of primes p, q with $p' = \frac{p-1}{2}, q' = \frac{q-1}{2}$ also primes (each of p, q is a so-called strong prime). Shoup gives a security proof in a security model under well studied assumptions. In Shoup's scheme calculations are performed modulo $n' = p'q'$, the order of the subgroup. The signing key d is shared by an adaptation of the usual method of choosing a random polynomial g modulo n' of degree $k-1$ with $d = g(0)$. Signers $P_1, \ldots, P_l$ are given shares $g(1), \ldots, g(l)$ respectively. Since d can be written as a suitable linear combination of any k of $g(1), \ldots, g(l)$, a normal RSA signature m^d may be calculated as a product of suitable powers of any k of the partial signatures $m^{g(1)}, \ldots, m^{g(l)}$. That such partial signatures have been calculated honestly may be checked by an adaptation of the Chaum-Pederson [8] proof of equality of discrete logarithms.

So far we have described (at a high level) Shoup's threshold RSA signature scheme. Shoup's paper [14] contains two distinct variant schemes. It is quite straightforward to extend these schemes to support blinding, resulting in threshold blind signature schemes. (In fact, Shoup's second scheme already involves a blinding factor one half of the time.) The threshold value k could be set to 1, in which case any single sub-mint can create valid signatures, and the private key shares $g(i)$ are simply copies of the original private key. Higher threshold values require more sub-mints to be involved – this increases the security of the private key, but requires more (total) computation and communication.

One very nice feature of the Shoup-style threshold blind RSA scheme is that no interaction between sub-mints is required to create signature components; the user can interact separately and directly with the different sub-mints and combine the signature components that he receives for himself. However randomized blinded signatures would require the sub-mints to agree upon the randomizing value x.

Xu and Chen [15] propose another method of threshold RSA signatures but do not provide a security proof. Instead of working in the group of squares modulo n they work in $\mathbf{Z}_w$ for some prime $w > n$. Lagrange interpolation is used to express d as a sum of shares modulo w. To obtain the signature as a

product of partial signatures modulo n a correction factor of a suitable power of an additional 'share' $m^r \pmod n$ is required. This is done by an exhaustive search of k possibilities using public information about a reference signature of a fixed Blank m_0. Again a randomized signature would require the Signers to agree upon the value x.

There is a good deal of further literature on blind threshold signatures, particularly for the discrete-log (rather than the RSA) setting. Interpreting this literature would require extensive project time. One immediate point, already made by Shoup in [14], is that discrete-log schemes tend to require interaction between the threshold signers, whereas the RSA-based scheme does not. In any case, using a discrete-log based scheme seems unnecessary given the availability of a simple and secure blind variant of Shoup's threshold RSA signature scheme.

4.3 Future directions

4.3.1 Are there protocols/algorithms besides Chaum which are more suited?

This depends on the exact requirements. Chaum's scheme is on-line, whereas "modern" schemes are off-line and offer stronger double spending detection and traceability guarantees. However, these latter schemes require the users to have public keys and register them as part of a PKI. Such schemes may also require significant additional implementation effort, since they are based on more advanced computational primitives than the simple RSA-style arithmetic used in Chaum's scheme.

4.3.2 Receipts

This is rather easily done if one assumes a PKI is deployed to support the system. A digital signature can then be used by a merchant to create a receipt for any coins received from a spender. The phrase "the issuer can detect if receipts were exchanged and refuses redemption otherwise" in the stated requirement seems to suggest that the mint can link the receipt to a coin received from a merchant, but also link the receipt to a particular spender and/or merchant. This then seems to sit in contradiction to the anonymity requirement that the mint not be able to tell who spent which coins and on what. At the minimum, the messages being signed would need to be carefully constructed, avoiding signing the spender's identity, for example. It would be important before proceeding to explore further why this feature might be required at all – the security properties required of an e-cash scheme imply that such receipts are not needed anyway in order to ensure smooth operation of the scheme.

4.3.3 Offline tokens

Double-spending is inherent in any e-cash scheme, since coins are just bits and are trivial to copy. So some form of double-spending detection must be implemented to deter malfeasors in any scheme. Then the feasibility of any e-cash

scheme depends on the ease with which double spending can be detected, and the degree of traceability which results from successful detection. The basic split in classification for e-cash schemes is between on-line and off-line schemes. OpenCoin has implemented an on-line scheme, in the original spirit of Chaum's proposal. Off-line schemes are certainly feasible, and there are many to choose from, but implementing one would require significant re-engineering of the current on-line implementation. It would also seem to require strong user identities and PKI for users.

4.3.4 Transferable tokens

The idea here would seem to be that, once a coin is received by party B from party A, it can be passed on as payment for another service by B (to C, say). The problem with this approach in standard (off-line) e-cash is that nothing can prevent B spending the coin multiple times and through this framing A as a double spender. In an on-line scheme, it is in B's interests to redeem the coin from A with reasonable haste, to prevent A from double spending the coin in an undetectable way (nothing prevents A from doing this even after the coin has been transferred to B). Most current e-cash schemes do not support this kind of transferability property. However, some research has been done into schemes offering transferability. We could certainly study this literature more closely if the project has a follow-on phase. One paper of relevance is [13].

A simple method trading transferability for anonymity would be for A simply to provide a signature of transfer of ownership of the coin for B. B can lodge this signature with the bank, to prevent A from double spending the coin. This receipt of transfer need not identify the recipient of the coin, only that the coin has been transferred by A to *someone*. Potentially, A can also lodge a corresponding receipt of transfer from B at the bank, to prevent B framing A as a double spender; however, this only seems important in an off-line scheme. This approach would require the bank to be online (whether the starting e-cash scheme was on-line or not) and would potentially compromise the anonymity of the scheme. It would also require the users to have signing keys and therefore a user PKI.

It may be possible to use optimistic fair exchange techniques together with an offline Trusted Third Party to preserve anonymity (except in the event of dispute) for coin transfers of this type. This would require further design work and security analysis. Similar ideas seem to arise in the recent paper [3] – the connection may be worth investigating more closely.

4.3.5 Conditional anonymity

This might be enabled using an additional TTP (representing the law enforcement agency) and verifiable encryption techniques. Essentially, when the user blinds a coin, he provides to the mint along with the blinded coin a verifiably encrypted version of the coin's serial number, using the TTP's public key to perform the encryption. The verifiable nature of the encryption would allow

the issuing bank to be sure that the user has properly encrypted the serial number, but without revealing that serial number (otherwise anonymity is broken). What we need in order to achieve this is a cryptographic proof that the string that has been encrypted is the same as the serial number of the coin that has been blinded. This proof will be scheme-specific and could be difficult to construct. It may be even better from a non-repudiation perspective if the user verifiably encrypts a signature of the serial number. But this could make the proof of correctness even more complicated.

An alternative would be to provide a verifiable encryption of the blinding factor itself – again, any such proof would be scheme-specific and it would likely require careful research to obtain a working scheme. It should also be noted that this facility (of course) runs counter to the desire of having fully anonymous e-cash.

References

[1] M. Abe and E. Fujisaki. How to date blind signatures. In *Advances in Cryptology – ASIACRYPT96*, LNCS Vol. 1163, pp. 224–251, Springer-Verlag, 1996.

[2] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. J. Cryptology 16(3), pp. 185–215 (2003)

[3] J. Camenisch, A. Lysyanskaya and M. Meyerovich. Endorsed e-cash. In *IEEE Symposium on Security and Privacy 2007*, pp. 101–115, IEEE Computer Society Press, 2007.

[4] T. Cao, D. Lin and R. Xue. A randomized RSA-based partially blind signature scheme for electronic cash. Computers and Security 24(1), pp. 44–49 (2005).

[5] Z. Cao, H. Zhu and R. Lu. Provably secure robust threshold partial blind signature. Science in China Series F: Information Sciences 49(5), pp. 604–615 (2006).

[6] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO82*, pp. 199–203, Plenum, 1983.

[7] D. Chaum, A. Fiat and M. Naor. Untraceable Electronic Cash (Extended Abstract). In *Advances in Cryptology – CRYPTO'88*, LNCS Vol. 403, pp. 319–327, Springer-Verlag, 1990.

[8] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO92*, LNCS Vol. 740, pp. 89–105, Springer-Verlag, 1993.

[9] H.-Y. Chien, J.-K. Jan and Y.-M. Tseng. RSA-based partially blind signature with low computation. In *IEEE 8th International Conference on Parallel and Distributed Systems*, pp. 385–389, IEEE press 2001.

[10] H.-Y. Chien, J.-K. Jan and Y.-M. Tseng. Partially Blind Threshold Signature Based on RSA. INFORMATICA 14(2), pp. 1–12, 2003.

[11] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In *Advances in Cryptology – CRYPTO91*, LNCS Vol. 576, pp. 457–469, Springer-Verlag, 1992.

[12] C.I. Fan, W.K. Chen and Y.S. Yeh. Randomization enhanced Chaums blind signature scheme. Computer Communications 23(17), pp. 1677–1680, 2000.

[13] J.K. Liu, S.H. Wong and D.S. Wong. Transferable E-Cash Revisit. In *Security and Privacy in the Age of Ubiquitous Computing: : IFIP TC11 20th International Information Security Conference*, pp. 171-188, Springer-Verlag, 2006.

[14] V. Shoup. Practical Threshold Signatures. In *Advances in Cryptology – EUROCRYPT 2000*, LNCS Vol. 1807, pp. 207–220, Springer-Verlag, 2000.

[15] Q.-L. Xu and T.-S. Chen. An efficient threshold RSA digital signature scheme. Applied Mathematics and Computation 166(1), pp. 25–34, 2005.