Flexibly-Configurable and Computation-Efficient Digital Cash with Polynomial-Thresholded Coinage

Alwyn Goh[1], Kuan W. Yip[2], and David C.L. Ngo[3]

[1] Corentix Laboratories, B-19-02 Cameron Towers, Jln 5/58B,
46000 Petaling Jaya, Malaysia. alwyn_goh@yahoo.co.uk
[2] Help Institute, BZ-2 Pusat Bandar Damansara,
50490 Kuala Lumpur, Malaysia
[3] Faculty of Information Science & Technology, Multimedia University,
75450 Melaka, Malaysia

Abstract. This paper describes an extension of the Brands protocol to incorporate flexibly-divisble k-term Coins via application of Shamir polynomial parameterisation and Feldman-Pedersen zero knowledge (ZK) verification. User anonymity is preserved for up to k sub-Coin Payments per k-term Coin, but revoked for over-Payments with (k+1) or more sub-Coins. Poly-cash construction using only discrete logarithm (DL) or elliptic curve (EC) operations enables efficient implementation in terms of the latter; which constitutes an advantage over previous divisble Coin formulations based on quadratic residue (QR) binary-trees, integer factorisation (IF) cryptography or hybrid DL/IF. Comparative analysis of Poly-cash and previous protocols illustrates the advantages of the former for operationally realistic Coin sub-denominations. The advantage of Poly-cash in terms computational overhead is particularly significant, and facilitates implementation on lightweight User Purses and Merchant Payment-terminals. Configurable k-divisibility is also an important consideration for real-world applicability with decimal currency denominations, which is not well addressed by the binarised values of QR-tree divisible Coins.

1 Introduction

Digital cash protocols—specifying interactions between distinct User, Bank and Merchant entities—typically emphasise anonymous Payment transactions and feature Coin data-structures which can be recognised as authentic. These attributes are characteristic of physical currency, the *goodness* of which can be established without User identity being an operational issue. The notion of User anonymity in a digital cash context is usually modified to result in preservation only in the event of legitimate Coin usage. Conditional anonymity allows for offline Coin verification—with respect having been issued by a particular Bank, without compromising User identity and without the necessity for Merchant-to-Bank connectivity—during a User-to-Merchant Payment. Merchants and Banks are, however, protected via detection of User fraud during subsequent Merchant-to-Bank Deposits. The conceptual framework for digital cash protocols was defined by the groundbreaking research of Chaum [1] and Brands [2, 3], the latter of which constitutes the base-formulation (with single-term non-divisible Coins) for featured protocol.

The concept of Coin-divisibility (with multiple sub-Coins per Coin) was first formulated by Okamoto-Ohta [4, 5], and is motivated by the reduced overheads arising from amortisation of computationally-expensive Withdrawals. Okamoto divisi

A. Lioy and D. Mazzocchi (Eds.): CMS 2003, LNCS 2828, pp. 181–193, 2003.

bility is based on QR/IF cryptography, with the Coin a binary-tree of successively computed QRs. Such a data-structure—subsequently also used in the Chan-Frankel-Tsiounnis [6] protocol—has 2^k leaf-nodes for a tree of height k, allowing sub-Coins of binarised fractional value $\left(\frac{1}{2}\right)^k$ for $k \in [1, n]$. QR-tree Coinage is extremely efficient for large k values, but would result in relatively high overheads for smaller denominations due to the necessity for long QR parameter-lengths. The Ferguson protocol [7] also features Coin divisibility based on the Rivest-Shamir-Adleman (RSA) formulation, and would therefore require similarly high overheads.

We outline a divisible cash protocol with flexibly configurable k-term Coins—with sub-Coins of relative fractional value $\frac{1}{k}$—specified via Shamir [8] polynomials and Feldman-Pedersen [9, 10] verification. The proposed divisibility mechanism can be expressed in terms of DL or EC finite-field operations, the latter of which results in significantly reduced overheads. This is a major advantage over the QR/IF basis of previous divisible cash protocols. The resultant k-term Coins and extended Brands operational framework is both efficient and versatile, for instance being straightforwardly supportive of practical (k = 10) *dime* or (k = 20) *nickel* denominations.

2 Review of Single-Term Brands Protocol

2.1 Bank/User Setup

The Brands protocol can be described in terms of the following processes:-
(1) *Bank Setup*: establishment of common computational environment
(2) *User Setup*: account establishment and individual licensing for Coin-generation
(3) *Withdrawal*: generation of anonymised Coin, resulting in User account-debit
(4) *Payment*: transfer of transaction-committed Coin from User to Merchant
(5) *Deposit*: submission of Coin to Bank, resulting in Merchant account-credit
as illustrated below:

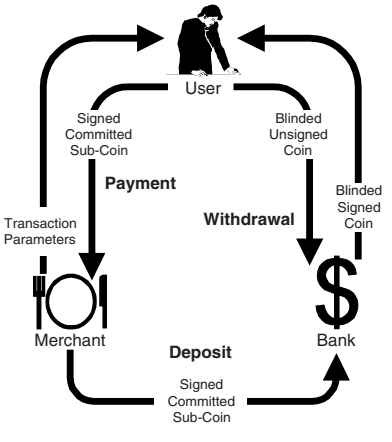

Fig. 1. Digital Cash Withdraw-Pay-Deposit Cycle

Brands digital cash—in common with other DL formulations—can transcribed in terms of EC finite-field operations, thereby leveraging the latter's order-of-magnitude advantage in terms of computation, storage and communications for equivalent-security parameterisations. The Brands scheme in particular requires publication of the following environmental parameters: (1) primes (p, q) with q | p–1 as previously defined, (2) basepoints $(\mathbf{g}, \mathbf{g}', \mathbf{g}'') \in E_p^{a,b}$ on the elliptic curve defined over Z_p with number of points divisible by large prime q, and (3) public component of Bank key-pair (x, **y** (= x**g**)); during Bank Setup.

Individual User Setups episodes can subsequently proceed with the generation of individual key-pairs (μ, **I** (=μ**g**′)); with the public components submitted to the Bank, associated with specific User accounts and signed (using Bank private-key x) to create Coin-generation licenses of form **z** = x(**I** + **g**″). In the Brands framework the different base-points serve various functions ie **g** for blind-signature generation and (**g**′, **g**″) for representation of the Coin data-structure.

2.2 User-to-Bank Withdrawal

The simultaneous requirements for verifiable Coin-authenticity and conditional User-anonymity is ensured by the Bank generating a blinded Schnorr signature for each Coin, as outlined below:-

Table 1. User-to-Bank Withdrawal

	Bank B		*User U*
1	Generate random w Compute Schnorr (**a**, **b**)	(**a**, **b**) →	
2			Generate blinding (s, u, v) Compute Coin **A** and (**z**′, **a**′, **b**′) Generate secret-splitting (x′, x″) Compute Coin **B** = x′**g**′ + x″**g**″
		← c	Compute Coin challenge (c′, c)
3	Compute r = cx + w mod q	r →	
4			Verify • r**g** = c**y** + **a** • r(**I** + **g**″) = c**z** + **b** Compute r′ = ru + v mod q

The Schnorr signature protocol (in common with other EC/DL formulations) requires a secret signer-determined randomisation w; which is subsequently used to compute (**a**, **b**) = (w**g**, w(**I** + **g**″)), the second component of which is User-specific. The subsequent blinding—by the signature verifier (User)—usually requires four parameters, two each for message blinding and manipulation of the signer (Bank) response to a verifier-issued challenge. The choice of (**I** + **g**″) as the Schnorr message necessitates the application of User private-key μ as one of the blinding parameters, along with random Coin-specific (s, u, v). This allows the computation of (**A**, **z**′) = (s(**I** + **g**″), s**z**)

and ($\mathbf{a}'$, $\mathbf{b}'$) = (u**a** + v**g**, su**b** + v**A**). Parameter **A** can be regarded as the User-specific part of the Coin, while ($\mathbf{z}'$, $\mathbf{a}'$, $\mathbf{b}'$) are components of the (as yet incomplete) signature.

Encoding of the User secret into the Coin is established randomly generated factors (x', x'') and **B**. The latter constitutes a User-to-Bank commitment on the specific Coin, and is later used for the verification of the challenge-response pair during Payment. This is followed by computation of unblinded $c' = h(\mathbf{A}, \mathbf{B}, \mathbf{z}', \mathbf{a}', \mathbf{b}')$ and blinded $c = c'_u{}^{-1} \bmod q$ challenge parameters, both of which are Coin-specific and the latter of which is then presented to the Bank for blind-signature affixation. All of computations thus far constitute pre-transaction operations which can, in fact, be executed (and the generated parameters stored) prior to an actual Withdrawal episode resulting in User account-debit. This speeds up Coin-generation to a significant extent, which consequently only requires the User-to-Bank challenge-response sequence in the last two steps of Table 1 for satisfactory conclusion.

A proper response r to blinded challenge c would require knowledge—presumed as restricted to the Bank—of private-key x and transaction parameter w. Such a response for blinded message (**z**, **a**, **b**, c) can be verified using Bank public-key **y** and enables construction of unblinded signature $\sigma(\mathbf{A}, \mathbf{B}) = (\mathbf{z}', \mathbf{a}', \mathbf{b}', r')$, following which the Bank can be safely authorised to execute the account-debit. The interpretation for $\sigma(\mathbf{A}, \mathbf{B})$ as a blind-signature can be seen from the Bank not knowing (**A, B**), in conjunction with the dependence of third-party (Merchant) verification condition ($r'\mathbf{g}$, $r'\mathbf{A}$) = ($c'\mathbf{y} + \mathbf{a}'$, $c'\mathbf{z}' + \mathbf{b}'$) on Bank public-key **y**. Coin $\chi = (\mathbf{A}, \mathbf{B}, \sigma(\mathbf{A}, \mathbf{B}))$ is hence verifiably authentic, while also entirely protective of User anonymity.

2.3 User-to-Merchant Payment

Coin χ can subsequently be used for Payment as follows:-

Table 2. User-to-Merchant Payment

U	Compute challenge $c = h(\mathbf{A}, \mathbf{B}, \mathbf{I}_m, T)$
	Compute response (r', r'') for c
	$\downarrow \chi(\mathbf{A}, \mathbf{B})$, T, ($r'$, r'')
Merchant M	Verify Bank signature $\sigma(\mathbf{A}, \mathbf{B})$
	Compute challenge c
	Verify response $r'\mathbf{g}' + r''\mathbf{g}'' = c\mathbf{A} + \mathbf{B}$

Note the Payment-specific non-interactive challenge c—which incorporates Merchant ID $\mathbf{I}_m$ and external specifier (ie timestamp) T—the correct response for which is (r', r'') = ($c(\mu s) + x'$, $cs + x''$). The response: (1) encodes μ and Coin-specific blinding factor s, while (2) concealing s; thereby facilitating recovery of the User secret (and consequently identity **I**) with two or more of the challenge-response pairs submitted during the subseqeuntly described Deposit. User anonymity is therefore conditional on proper use ie not more than one Payment per Coin.

2.4 Merchant-to-Bank Deposit

Deposit is simply Merchant-to-Bank submission of the Coin and Payment-specific commitment $\chi'(\mathbf{A}, \mathbf{B}) = (\chi(\mathbf{A}, \mathbf{B}), (c, r', r''))$, as outlined below:

Table 3. Merchant-to-Bank Deposit

M	$\downarrow \chi'(\mathbf{A}, \mathbf{B})$
B	Verify own signature $\sigma(\mathbf{A}, \mathbf{B})$ Verify response $r'\mathbf{g}' + r''\mathbf{g}'' = c\mathbf{A} + \mathbf{B}$ Check for over-use of $\sigma(\mathbf{A}, \mathbf{B})$ If detected, then:- • Compute $\mu = \frac{r' - \rho'}{r'' - \rho''}$ • Compute $\mathbf{I} = \mu\mathbf{g}'$ for identification

with (ρ', ρ'') some previously catalogued response in the Bank database, threby enabling detection of double spending. There are three types of fraud which can be attempted on a particular Coin, with multiple responses that are: (1) identical, (2) non-identical but non-verifiable, and (3) non-identical and verifiable. The first two cases can probably be attributed to Merchant error or fraud, with User non-liability in (2) due to malformation of the commitments. Case (3) is User double-spending since valid commitments can only be computed by the User, whose anonymity is the revoked via recovery of key-pair $(\mu, \mathbf{I})$ from multiple Payment-specific responses.

3 Review of Polynomial Secret-Sharing

Our scheme is based on the division of the Brands' Coin to multiple sub-Coins via polynomial secret-sharing (SS). Initial setup requires selection of a configurable *threshold* (k) so that the secret—in our particular case the User-identity—can be be *split* into an arbitrary (n) number of shares, with secret-reconstruction necessitating possession of k+1 (or more) shares. On the other hand, possession of k (or less) shares is cryptographically equivalent to knowing nothing about the divided secret, hence preserving User anonymity.

There are various encoding schemes for such divided secrets, with the most straightforward (due to Shamir) based on polynomial interpolation over a finite field. In such a scheme a k-th degree polynomial $f(x) = \sum_{i=0}^{k} b_i x^i \bmod q$ is used to encode secret $f(0) = b_0$, with the remaining k coefficients $b_i \in Z_q$ (for $i \neq 0$ and q prime) randomly generated and kept secret by the User. The Coin-specific polynomial is then recoverable via Lagrange interpolation given k+1 distinct (x, f(x)) coordinate pairs on the polynomial-defined curve, enabling polynomial reconstruction $f(x) = \sum_{i=0}^{k} f(x_i) \left[\prod_{j=0, \neq i}^{k} \frac{x - x_j}{x_i - x_j} \right] \bmod q$ and subsequent recovery of secret f(0).

Polynomial-based SS is attractive due to the inherent threshold property preventing the manipulation of k (or less) (x, f(x)) shares for any useful information on the encoding polynomial.

Shamir SS does, however, assume honest share allocation by the User, which is operationally unrealistic. The disclosed shares must therefore be *verifiable*, thereby allowing receivers to validate share association with the secret polynomial. This can be established via Pedersen verified SS (VSS) which represents both the secret and its shares in terms of EC/DL images, thereby resulting in ZK share-verification. The Pedersen formalism—expressed in terms of EC operations with respect $\mathbf{g} \in E_p^{a,b}$ —requires prior disclosure of polynomial *commitments* $\beta_i = b_i\mathbf{g}$, thereby allowing ZK polynomial evaluation via $f(x)\mathbf{g} = \sum_{i=0}^{k} x^i\beta_i = \sum_{i=0}^{k} b_i x^i\mathbf{g}$. This enables a Merchant to verify the legitimacy of an individual share (sub-Coin) with respect a Coin-specific secret, without acquiring cryptographically useful information on the Coin and consequently the User. This results in User anonymity preservation unless there are if k+1 (or more) distinct sub-Coins (Payments) corresponding to a k-term Coin.

Polynomial SS is an elegant mechanism with which to implement Coin divisibility into the Brands digital cash protocol. Note this results in both divisibility and conditional-anonymity mechanisms being based on EC operations, in contrast to previous formulations with Coin divisibility based on QR/IF cryptography. Our approach also enables interpretation of Brands conditional-anonymity—which is revoked from two distinct challenge-response pairs—as a special case with linear (k=1) polynomials.

4 Poly-Cash: k-Divisible Polynomial-Based Coins

Incorporation of polynomial divisibility requires requires extending the above-outlined Withdrawal, Payment and Deposit operations—for single-term Brands Coins—as follows:

4.1 Coin Withdrawal

Coin-specific polynomials f(x) (of degree k) can be used to encode User identity ie $b_0 = \mu$, with the remaining coefficients randomly generated and kept secret by the User. k-term Coin-generation therefore entails generation of polynomial coefficients and their EC images ie $\left(b_i, \beta_i \left(= b_i\mathbf{g}''\right)\right)$ for i = 0…k., with integration into the Brands formulation via specification of Coin-parameter $\mathbf{B} = h\left(\prod_{\forall i} \beta_i\right)$. This constitutes a User-to-Bank commitment on the Coin-specific polynomial which (if over-spent) can subsequently be used for secret-recovery.

Withdrawal and Coin-generation then proceeds as follows:

Table 4. Coin Withdrawal

	B		*U*
1	Generate random w Compute Schnorr (**a**, **b**)	(**a**, **b**) →	
2			Generate blinding (s, u, v) Compute Coin **A** and (**z**′, **a**′, **b**′) Generate secret-splitting (x′, x″) Compute Coin **B** from β_i Compute Coin challenge (c′, c)
		← c	
3	Compute r = cx + w mod q		
		r →	
4			Verify • **rg** = c**y** + **a** • r(**I** + **g**″) = c**z** + **b** Compute r′ = ru + v mod q

with a heavier computation overhead (by a factor of $\frac{k}{2}$) for the computation of **B**. The first two steps are—in common with the base-formulation of Table 1—precomputable, the result of which is improved efficiency compared to Withdrawal of k distinct single-term Coins.

4.2 Sub-coin(s) Payment

Note the essential similarity of Tables 1 and 4, which differ only in the construction of the **B** component of Coin χ. The subsequently outlined Payment and Deposit, on the other hand, must now deal with sub-Coins—of value $\frac{1}{k}$ in relation to k-term Coin χ—as the fundamental transactional unit. The basic concept is to commit each sub-Coin Payment via disclosure of a Shamir polynomial share, which can then be ZK-verified (by the Merchant) using the Pedersen formalism. This necessitates User disclosure of Coin χ and polynomial-commitments β, both of which are verifiable via σ(**A, B**). A single share of the encoded secret is hence (c, r (= f(c))), which is verifiable via $r\mathbf{g}'' = \sum_{i=0}^{k} c_i\beta_i$. Note that knowledge of polynomial-coefficients b is required for proper construction of verifiable responses, thereby restricting Payment to the legitimate User.

Payment then proceeds as follows:

Table 5. Sub-Coin Payment

U	Compute sub-Coins $(c,r)_j$
	$\downarrow \chi, \beta_i, (T,r)_j$
M	Verify Bank signature σ(**A, B**)
	Verify Coin polynomial β_i
	Verify sub-Coin shares $(c,r)_j$

and can be repeated to transfer n sub-Coins—for n = 1…k and with a relative value of $\frac{n}{k}$—via computation of n challenge-response pairs of form $(c,r)_j$ for j = 1…n. Note the incorporation of non-interactive challenges $c_j = h\left(\mathbf{A}, \mathbf{B}, \mathbf{I}_m, T_j\right)$—with $\mathbf{I}_m$ and T_j as previously specified—resulting in a simplified transactional framework. Payment using multiple sub-Coins divided from a k-term Coin (with reuse of all steps involving χ and β) is also more efficient than the alternative of several equally-denominated single-term Coins. Successful Payment results in Merchant possession of User-commitment on n sub-Coins of form $\chi'(\mathbf{A},\mathbf{B}) = \left(\chi, \beta_i, (c,r)_j\right)$, the Deposit of which is subsequently described. The computation complexity for a single sub-Coin only involves generation of the challenge-response pair, which requires $\frac{k(k+1)}{2}$ modular multiplications with *k* a small (by cryptographic standards) integer. An average case of $\frac{k}{2}$ sub-Coins per Payment would therefore result in an overhead of $\frac{k^2(k+1)}{4}$ modular multiplications.

4.3 Sub-coin(s) Deposit

Digital cash protocols are specifically designed to preserve User-anonymity in the event of legitimate Coin usage, which in our formulation is specified by the Bank not having a priori knowledge of Coin details $\left(\mathbf{A}, \mathbf{B}, \beta_i\right)$ and additionally by the number of sub-Coins being below the recovery threshold. Over-Payment associated with a k-term Coin (by a particular User) is detectable by the Bank via scrutiny of χ'(**A**, **B**) Deposit data for possible accumulation of k+1 (or more) distinct and verifiable shares. These $(c,r)_j$ shares can then be combined (using Lagrange interpolation) by the Bank, resulting in knowledge of the Coin-specific f(x) and subsequently identification of the overspending User via (μ, **I**) computation.

The following Deposit process:-

Table 6. Sub-Coin Deposit

M	$\downarrow \chi'$ (**A, B**)
B	Verify own signature σ(**A, B**) Verify Coin commitments β_i Verify sub-Coins $(c, r)_j$ Check for over-use of (**A, B**) If detected, then:- • Recover Coin f(x) • Compute (μ, **I**) for identification

leads (barring detection of fraud) to Merchant account-credit by $\frac{n}{k}$ the value of Coin χ. Note the emphasis on a posteriori detection of User fraud; which contrasts with the in situ prevention of Merchant or third-party fraud, both of which are effectively ruled out by the necessity for verifiable sub-Coin commitment. The Brands protocol can actually be extended to include User-side Observer modules for User fraud prevention (as opposed detection) at the expense of additional computation. The incorporation of VSS-based divisibility as outlined in this document is designed to ensure Coin-structural compatibility with respect the Brands framework, thereby allowing for Observer-based fraud preventive measures.

The featured protocol does, however, allow the Bank to establish a posteriori sub-Coin (same Withdrawal, different Payments) level linkages; even when User anonymity is not revoked. This trait is also present in the Brands and Okamoto formulations, but not in the Chan et al protocol due to its elimination and downward-movement of the User Setup procedure into the Withdrawal operation. Such a strategy is somewhat at odds with one of the important motivations for divisible cash ie to reduce per-Payment overheads related to Withdrawal, which is particularly heavy due to the complexity of conditional anonymity enablement and User identity encoding into individual Coin. Note that our scheme can also be rendered unlinked via use of Coin-specific identities, as in the Chan et al formulation. The most practical resolution of this issue is, however, operational in nature via periodic refreshment (via User Setup repetition) of identity vector (μ, **I**, **z**).

5 Comparative Analysis

We compare the performance of the above-outlined EC-based k-term scheme with *equivalent-security* [11] parameterisations of previously published protocols. This amounts to 160-bit moduli-lengths for our EC formulation being equivalent (at currently accepted equivalent security levels) to 1024-bit moduli-lengths for DL-based schemes ie Brands, Okamoto and Chan et al. Equivalent-security considerations also requires parameter-length adjustments ie standardisation of Okamoto's (P, Q, N, n, b) to (512, 512, 1024, 1024, 128) and an upgrade of Chan-Frankel-Tsiounnis' (P, Q, N, p, q) to (512, 512, 1024, 1024, 512).

Comparison of DL and EC overheads requires cross-calibration of the two most significant operations for each formalism ie DL multiplication (Mult) vs EC point addition (PA) and DL exponentiation (Exp) vs EC scalar multiplication (SM). The computational costs of these operations is analysed in [11, 12] and can be summarised as follows:-

- *Mult*: $\lg^2 p$
- *Exp* (via Repeated Squaring): $\lg q \cdot \lg^2 p$
- *PA*: 3.5 Mults and one modular-inversion, which costs $\lg^2 p$
- *SM* (via Addition-Subtraction): lg p doublings/PAs and $\lg \frac{p}{2}$ PAs

Note the EC operations presume a shorter moduli-length. This leads to evaluation of the relative Mult-to-PA overheads [12] as approximately 9 at the equivalent setting of (DL, EC) = (1024, 160)-bit, with an increase to 23 for the more rigorous (2048, 200)-bit [11] setting. Similar analysis establishes a Exp-to-SM ratios [12] of approximately 6 at the *regular*-security setting, with an increase to 15 at the *high*-security setting. The relatively small increase in the EC moduli-lengths can be attributed to its exponential effect on computational security, as opposed increases in DL/IF moduli-lengths resulting in much less dramatic sub-exponential enhancements.

Our analysis examines the computation, storage and communications overheads in terms of the leading-order significant operations (mostly Exp and SM); and presumes pre-computation whenever allowed by the respective protocols. We also adjust for the continuous divisibility of QR-tree based Coins via assignment of an average height h (=lg k). The real-time overheads of the (1) Okamoto, (2) Chan et al, (3) Brands and (4) Poly-cash schemes are summarised as follows:-

Table 7. Computation, Storage and Bandwidth Overheads of Various Schemes

		Computation	*Storage*	*Communications*
User Setup	1, 2	$\lg \lvert P \rvert \cdot \lvert P \rvert^4 \cdot$ Exp(P)	2|N|+|P|+|Q|	$\lg \lvert P \rvert \cdot \lvert P \rvert^4$
	3	2 Exp (p)	|p|	|p|
	4	2 SM (p)	|p|	|p|
Withdrawal	1	2 Exp (N)	2|N|+|P|+|Q|	2|N|
	2	12 Exp (N)	2|p|+|q|+2|P|	3|p|+2|q|
	3	6 Exp (p)	5|p|+|q|	2|p|+2|q|
	4	6 SM (p)	6|p|	4|p|
Payment	1	(h+1) exp (N)	|N|*(h+1)	|N|*(h+1)
	2	(h+1) exp (N)	|N|*(h+1)	|N|*(h+1)
	3	2 Mult (q)	3|q|	3|q|
	4	k(k+1) Mul (p)	k|p|	2k|p|

with respective Coin bit-lengths of:

1. 3|N|+|b|+|P|+|Q| (=424)
2. 2|p|+|q|+2|P| (=4636)
3. 5|p|+|q| (=960)
4. 6|p| (=960) for decimal (k=10) divisibility

Poly-cash can be seen to have the most efficient Setup and Withdrawal processes, with the most striking comparison in case being with respect Okamoto and Chan et al. This illustrates the usefulness of an EC-compatible divisibility mechanisms, as opposed the alternative of tree-node sub-Coins based on QR/IF cryptography. The Setup and Withdrawal processes are, in fact, essentially as efficient as an EC transcription of the Brands protocol, with negligible additional overheads arising from Coin divisibility. Note also the relatively modest sub-kbit size of the k-term Coin, thereby facilitating implementation of Coin-carrying Purses on lightweight handheld platforms.

Divisible Coins are specifically intended to amortise a particular Withdrawal overhead over multiple Payments, hence the most interesting analysis being from that viewpoint. Such schemes are predicated on the efficiency of Sub-Coin verification, which was the motivation for divisibility based on QR-trees. The intrinsic logarithmic efficiency and continuous divisibility of formulations based on QR-trees will eventually prevail for large divisibility (k) configurations. EC-based Poly-cash is linearly efficient with respect k; and is therefore expected to be advantageous for small values of k due to the previously discussed DL-to-EC operational ratios.

We found that our formulation to result in more efficient Payments—at least for practical divisibilities ie k~10—at the moderate (DL, EC) = (1024, 160)-bit equivalency setting. The *computation* and *communications* overheads for increasing *k* is illustrated below:

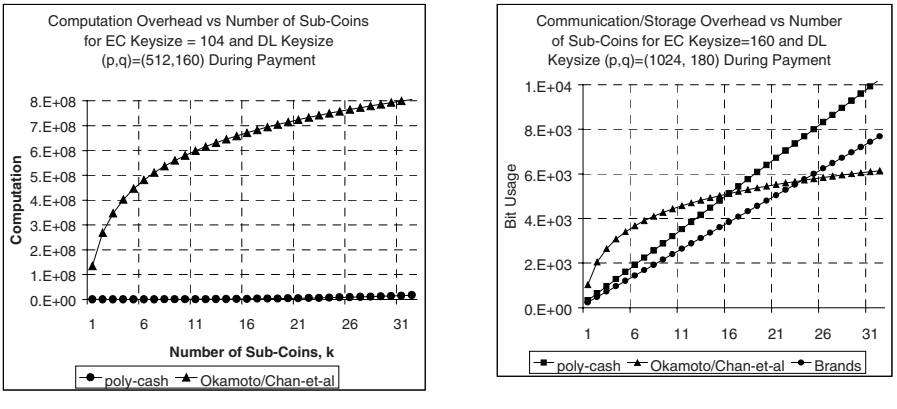

Fig. 2. (a) Computation and (b) communications overheads at moderate equivalent-security configurations

Note from Fig 2(a) that the equal-efficiency (crossover) point is not even on the graph, hence the lower computational overheads of Poly-cash even for fairly extreme divisibility settings. The crossover point in Fig 2(b), on the other hand, occurs around

k~16, which is still desirable as it exceeds the practical k=10 configuration with dime-sized sub-Coins. Comparison of communications overheads at the (2048, 200)-bit high-security equivalency setting, as follows:-

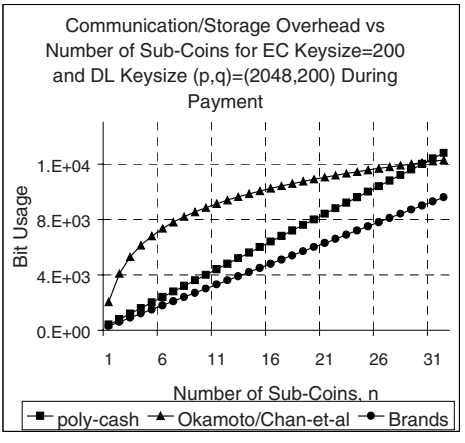

Fig. 3. Communications Overheads at High Equivalent-Security Settings

right-shifts the cross-over point to k~31, thereby demonstrating the relative efficiency of k=20 divisibility with nickel-sized sub-Coins.

6 Concluding Remarks

Polynomial thresholding is an elegant Coin-divisibility mechanism, and enables an efficient EC realisation impossible with the earlier QR-tree methodology. The combination of polynomial divisibility, an extended Brands operational framework and an EC implementation results in significant performance advantages (at equivalent-security parameterisations) compared to previous protocols. The flexible k-term divisibility is also preferable to the more rigid binarised denominations of the QR-tree formalism, especially with respect straightforward integration into existing trading and financial frameworks.

This and other digital cash protocols constitute viable alternatives to existing electronic payment systems, which typically require expensive Merchant-Bank connectivity during Payment. Offline Coin verifiability also lowers the operational overhead of Bank- side account management. User conditional anonymity also facilitates consumer privacy (without jeopardising fraud detectability) to an extent impossible with typical payment solutions.

References

1. D Chaum, A Fiat & M Noar. "Untraceable Electronic Cash". Crypto 88, Springer-Verlag Lecture Notes in Comp Sc (LNCS) (1988).
2. S Brands. "Untraceable Off-Line Cash in Wallets with Observers". Crypto 93, Springer-Verlag LNCS (1993).
3. S Brands. "An Efficient Off-line Electronic Cash System based on the Representation Problem". Tech Rep CS-R9323, CWI (1993)
4. T Okamoto & K Ohta. "Universal Electronic Cash". Crypto 91, Springer-Verlag LNCS (1991)
5. T Okamoto. "An Efficient Divisible Electronic Cash Scheme". Crypto 95, Springer-Verlag LNCS (1995)
6. Chan, A., Frankel, Y. & Tsiounnis, Y., "Easy come- easy go divisible cash", EuroCrypt 98, Springer-Verlag LNCS (1998)
7. N Ferguson. "Extensions of Single-Term Coins". Crypto 93, Springer-Verlag LNCS 773, pp 292-301(1993)
8. A Shamir. "How to Share a Secret". ACM Comms (1979)
9. P Feldman. "A Practical Scheme for Non-Interactive VSS". IEEE Symp Foundations Comp Sc (1987)
10. TP Pedersen. "Non-Interactive and Information-Theoretic Secure VSS". Crypto 91, Springer-Verlag LNCS (1991)
11. Menezes, A., "Comparing the security of ECC and RSA", at www.certicom.com, 11 Jan (2000)
12. P1363/D13, Draft Version 13, Standard Specification for PKC (1999)
13. WK Yip, "Divisible Digital Cash via Secret Sharing Schemes", Masters in Comp Sc Thesis, Universiti Sains Malaysia (2001)