

Mix-based Electronic Payments

Markus Jakobsson¹ David M'Raihi²

¹ Information Sciences Research Center, Bell Laboratories
Murray Hill, NJ 07974

`markusj@research.bell-labs.com`

² Cryptography and Security, Gemplus
34 rue Guynemer, F-92447 Issy-les-moulineaux Cedex
 `david.mraihi@ccmail.edt.fr`

Abstract. We introduce a new payment architecture that limits the power of an attacker, and improves the efficiency of the resulting scheme. Our proposed method defends against all known attacks, implements revocable privacy, and is well-suited for smartcard-based payment schemes over the Internet.

1 Introduction

Since the conception of anonymous payment schemes by Chaum [6] in the early eighties, a lot of attention has been given to new schemes for transfer of funds. Anonymity and its drawbacks, in particular, has been a busy area of research lately, giving rise to many solutions for balancing the need for privacy against the protection against abuse. However, all of these schemes have been based on the same basic architecture as the pioneering scheme, which we suggest may be an unnecessary limitation. By rethinking the underlying privacy and fund-transfer architecture, we show that a stronger attack model can be adopted, and costs curbed.

In order to strengthen the attack model, we make the following observation: It never pays to make any link of a chain stronger than the weakest link; what has not been given attention is that, in fact, it can be *damaging* to do so. Consider the following example: Assume we want to implement a payment scheme over the Internet. Due to the fact that the Internet is not an anonymous channel (although not authenticated either), we need to use some form of traffic anonymizer (e.g., a mix-network [5]) in order to obtain payer privacy against an eavesdropping adversary, no matter what kind of underlying payment scheme is employed. However, if the underlying payment scheme offers more privacy than what the traffic anonymizer implements, then an attacker of the system could potentially enjoy a higher degree of privacy than what is offered the honest user, e.g., by mailing in disks and asking to have the encrypted responses posted to newsgroups. Thus, if one link of the chain, corresponding to the privacy of one system component, is stronger than other, this potentially benefits an attacker without adding any value to the honest users.

We argue that it is unnecessary to employ a scheme with properties that can only be used by attackers. Moreover, by degrading the privacy of components

until all components deliver the degree of privacy of the weakest link, efficiency improvements can be found. Thus, we can strengthen the defense against attacks (some of which might not be known to date) at the same time as we make the system less costly, without making the system less attractive to its honest users. An example from the literature of this type of “double improvement” is the work of Jakobsson and Yung [15, 16, 17, 18], in which the *bank robbery attack* (in which the attacker obtains the secret keys of the banks) was introduced and prevented against, and the mechanisms employed to achieve this goal were used to improve the versatility and efficiency of the resulting scheme. By making sure that an attacker can never enjoy a higher degree of privacy than an honest user, and implementing mechanisms for a quorum of servers to selectively revoke privacy, we meet and surpass the degree of security of Jakobsson and Yung, while further reducing the costs for payments in many common payment situations.

Our scheme is, unlike other payment schemes with privacy, not based on blind signatures (or variations thereof), but its privacy is derived solely from the use of a mix-network, and authenticity of payments from the use of standard digital signature methods. It does not allow an attacker any higher degree of privacy than honest users, but all users enjoy the following degree of privacy: As long as no quorum of bank servers is corrupted or decides to perform tracings, and no payer or payee reveals how they paid or were paid, the bank servers can only learn the number of payments to and from each account during each time period (e.g., one month.) Moreover, as is appropriate, the transactions are independent in the sense that knowledge about one transaction gives no knowledge of another transaction (other than reducing the number of remaining possibilities.) The communication and computational costs for the different transactions are very similar to those of other schemes, but the amount of data stored by payers and merchants is significantly lower, making our suggested scheme particularly suitable for smartcard implementations.

Regarding the inherent requirements of any secure electronic commerce application, the usage of smartcards, both for their natural resistance to penetration and their mobile character, is an attractive idea. Nevertheless, a smartcard-based mobile payment application where all users (be they payers or merchants) could plug in their devices anywhere on a network to perform or collect payments or exchange e-mails anonymously is quite intricate due to the inherent and severe memory limitations of smart cards. We believe that the two main reasons why anonymous electronic payment schemes are not yet widely used are:

1. the issue of controlled anonymity (a common concern of banks and governments) has only recently been addressed, and
2. anonymous payment schemes based on the classical paradigm of coins are unsuitable for smart card implementations, by craving more memory than what has been technically achievable and affordable.

But let us for a moment take a step back from what features are desirable in a payment scheme, and focus on what is absolutely necessary. In order not to rely on any physical assumption, we need some form of authentication and

encryption. In order for the user not to have to put excessive trust in other entities, public key solutions has to be used. Therefore, let us call signature generation and encryption the minimal requirements (where the user performs all or some of the computation involved in performing the transactions.)

The most suitable and least expensive way of meeting these requirements would be to develop and use a very simple smartcard with a hardware accelerator for modular multiplications, and with a minimum of memory (since what is expensive in the design of smartcards is EEPROM and RAM). Such a product would be perfect for generating and verifying signatures (e.g. DSA [21] or RSA [23] signatures), and the use of a co-processor would allow very fast public-key certificate verification. It would offer the necessary operations required by all e-commerce applications. However, the use of such a smart card desperately clashes with existing solutions for anonymous e-commerce, since these require high amounts of storage, complicated protocols, and in many cases rely to some extent on tamper resistance.

The main advantage of our scheme is that we overcome the major drawbacks related to the original e-coin paradigm: using a mix-decryption scheme to provide controlled anonymity, we build a counter-based scheme where the participants do not need to store anything but their secret keys and a small amount of user-related information. In our setting, all participants could therefore be represented as owners of an inexpensive and simple smart-card of the type discussed above. We demonstrate how to build a payment scheme offering privacy, using only features of such a minimalistic cryptographic smartcard. Our solution offers:

- flexibility and plug-and-play ability to users: the same device enables home-banking, e-commerce or privacy-enhanced email applications,
- strong protection to users, issuers and governments by the use of controlled and balanced privacy.

2 Related Work

When the concept of electronic payments was introduced (see [6, 8]) there was a strong focus on *perfect* privacy. Lately, as possible government policies have started to be considered (e.g., [28]), and attacks exploiting user privacy have been discovered (e.g., [25, 15]), the attention has shifted towards schemes with *revocable* privacy.

In [2], Brickell, Gemmell and Kravitz introduced the notion of trustee-based tracing, and demonstrated a payment scheme that implemented computational privacy that could be revoked by cooperation between the bank and a trustee. The concept of *fair blind signatures* was independently introduced by Stadler, Piveteau and Camenisch [26]; this is a blinded signature for which the blinding can be removed by the cooperation between the signer (the bank) and a trustee. This type of signatures were employed in payment schemes in [4, 3], and a smart-card based variation was suggested in [20]. In [10, 9], methods were introduced allowing the trustee not to have to be on-line during the signature generation,

by employing so called *indirect discourse proofs*, in which the withdrawer proves to the bank that the trustee will later be able to trace.

In the above schemes, a cooperating bank and trustee were able to trace all properly withdrawn coins, but not coins obtained in other ways. The underlying attack model was strengthened by Jakobsson and Yung [15], by the introduction of the *bank robbery attack*, in which an attacker compromises the secret keys used for signing and tracing, or forces the signers to produce signatures using an alternate generation protocol; their proposed solution protects against this attack and permits *all* coins to be traced, and any coin to be successfully blacklisted after a short propagation delay. Also, the degree of trust needed was decreased, by the use of methods to assure that the tracing and signing parties do not cheat each other. This work was improved on in [16, 17, 18] by the distribution of the parties and the introduction of a minimalistic method for privacy revocation (i.e., verifying if a given payment corresponds to a given withdrawal.)

We achieve the same protection against attacks as the above constructions, and additionally, by shifting the model from a coin-based to an account-based model, make bank robberies futile: If the secret keys of the banks should be compromised, this only allows the attacker to *trace* payments, and *not* to mint money. Additionally, we meet the level of functionality introduced in [16, 17] in terms of methods for tracing, and distribution of trust and functionality. Finally, by the shift in architecture, we remove the propagation delay for blacklisting - in fact, we avoid having to send out blacklists to merchants altogether.

However, our scheme requires the payer to be able to communicate with a bank or clearing center for each payment. Also, it does not offer the same granularity of payments that the use of *challenge semantics* in [15, 16] could, but potentially requires multiple payments to match a particular amount (much like for many coin based schemes).

On the other hand, we are able to reduce the amount of information stored by the payer (who only has to store his secret key.) Moreover, the merchants neither have to store any payment-related information (after having verified its correctness), nor do they ever have to connect to the bank. (In some sense, one might say that the deposit is performed by the payer.) This allows for a very efficient implementation in a smartcard based Internet setting (where the smartcards are used by the payers to guarantee that only authorized users can obtain access to funds.)

The anonymity in our scheme is based solely on the use of a mix-network, a primitive introduced by Chaum [5]. We demonstrate an implementation based on a general mix-network (e.g., [27]), and state and prove protocol properties. We then consider the use of a particular, recently proposed type of mix-network [22, 1, 19], and discuss how its added features can further strengthen the payment scheme. The result is an efficient and practical payment scheme, well suited for Internet implementation, that prevents by all known attacks by ensuring that an attacker *never* can obtain any more privacy than an honest user is offered. The privacy is controlled by a conglomerate of banks and ombudsmen, a quorum of which have to cooperate in order to revoke it.

3 Model

Participants: There are five types of participants, all modeled by polynomial-time Turing machines: *payers*, *merchants*, *banks*³, a *transaction center*, and the *certification authority*. The payers and merchants have accounts with banks (but not necessarily the same banks); the transaction center processes transfers between accounts of payers and merchants, and is controlled by a conglomerate of banks; the certification authority issues certificates on all other participants' public keys, and may be controlled by banks and government organizations. The transaction center knows for each payer's public key the identity of the payer's bank. (If a user has several banks, he correspondingly has several public keys, one per bank.)

Trust: The payers and merchants trust that the banks will not steal their money. The payers trust for their privacy that there is not a quorum of dishonest, cooperating banks constituting a corrupt transaction center.

Computation: We base our general scheme on the existence of a mix-network. Such a scheme can be produced based on any one-way function. We also present a scheme using a recently proposed ElGamal based mix-network [19]. It uses the following, common assumption (the *hiding assumption* [7]): Let $p = 2q + 1$, for primes p and q , and let m, g be generators of a subgroup of order q . Then, the pairs (m, m^x, g, g^x) and (m, m^r, g, g^x) are indistinguishable, for random and unknown values $r, x \in Z_q$, $m, g \in G_p$.

4 Requirements

We will require the following from our scheme:

Unforgeability: It is not possible for a coalition of participants not including a quorum of bank servers to perform valid payments for a value v exceeding the value charged to these participants.

Impersonation safety: It is not possible for a coalition of cheating participants to perform a transaction resulting in an honest participant being charged more than what he spent.

Overspending blocking: It is not possible for a coalition of participants to perform payments that are accepted by merchants as valid for an amount exceeding their combined limits, or exceeding what they will be charged for.

Payment blocking: It is always possible for a user to block all payments from his account, going into effect immediately.

Revocability: The bank of any user can restrict the rights of the user to perform payments. If a user has no bank agreeing to a payment, the payment will not be performed.

³ It is possible to substitute some bank servers for ombudsman servers, whose aim it is to limit illegal tracing transactions by bank servers. These will not have to be trusted with funds, and only hold tracing keys. Due to a shortage of space, we do not elaborate on how exactly to implement these.

Framing-freeness: It is not possible for a coalition of participants not including an honest user to produce a transcript corresponding to a payment order from the honest user.

Uniform anonymity: The probability for any coalition of participants, not including a quorum of bank (and ombudsman) servers, to determine from whom a payment was made and to whom is non-negligibly better than a guess, uniformly at random from all possible pairs, given all pairs of payers and merchants corresponding to the input and output of the mix-network. It is not possible for a coalition of participants to obtain a higher degree of anonymity by forcing alternative protocols to be used.

Traceability: Any quorum of bank servers⁴ are able to perform the following actions:

1. Given identifying information about a payer (and possibly a particular payment transaction of the payer), establish the identity of the receiver of the corresponding payment(s).
2. Given identifying information about a merchant (and possibly a particular payment transaction to the merchant), establish the identities of the corresponding payer(s).
3. Given identifying information about a payer; a particular payment transaction from the same; a merchant; and a particular payment transaction to the same, establish whether these correspond to each other.

5 Paying and Tracing (General Version)

5.1 Setup

For each time interval between accounting sessions (in which the payment orders get decrypted and the merchants credited) a new public key may be used (we will elaborate on when this is necessary onwards.) These public keys can be broadcast beforehand. The corresponding secret keys are known only to the transaction center, and are kept in a distributed manner so that any quorum of mix-servers can calculate it.

Each party is associated with a public key, which is registered with the transaction center if the party in question is authorized to perform payments. Only this party knows the corresponding secret key, which is used to sign payment orders and possibly other data which needs to be authenticated. The signature scheme employed for this is assumed to be existentially unforgeable.

5.2 Paying

There are three phases of the payment scheme: *negotiation*, *initiation of payment*, and *completion of payment*. In the first, the payer and the merchant produce a description of the transaction; in the second, the transaction is being committed to and the payer's account debited; and in the third, the merchant's account is

⁴ As noted before, some of these may be controlled by an ombudsman.

credited accordingly. Note that the transaction becomes binding in the second phase already, so the transfer of the merchandise bought can be initiated right after the second phase has finished, and the third phase may be performed at a later point. This is a scenario rather similar to the credit card scenario, in which the payer commits to the transfer much before the merchant receives the funds.

1. Negotiation

In the negotiation phase, the payer and the merchant agree on an exchange, i.e., a description of the merchandise, how it will be delivered, the price of the merchandise, etc. We let m be a hashed-down description of this contract. Furthermore, the merchant gives the payer his account number a (which specifies the bank as well as identifies the merchants account in this bank) and a serial number s for this transaction. If the payment will constitute of several coins, a suite s_1, \dots, s_k of such serial numbers is given, each one representing one partial payment. A payment order o is of the form $m|a|s$. (If a suite of serial numbers is used, then a corresponding suite of payments orders o_1, \dots, o_k will be generated.)

2. Initiation of Payment

(a) The payer encrypts a batch of payment orders o_1, \dots, o_n using the public key encryption scheme of the *transaction center* (which corresponds to the mix-network). These payment orders not have to correspond to one transaction only, or have the same denomination, but may stem from multiple simultaneous transactions and be of different values. The result is a batch of encrypted payment orders, $\tilde{o}_1, \dots, \tilde{o}_n$. If different denominations are supported in the system, a description d is appended, describing what the values of the different transactions are. If only one denomination is supported, d is the empty string. The payer signs $\tilde{o}_1, \dots, \tilde{o}_n, d$ using his public key. The resulting signature is σ . The payer sends $\tilde{o}_1, \dots, \tilde{o}_n, d, \sigma$ to the transaction center.

(b) The transaction center verifies the signature σ . One of two approaches is taken in order to avoid replay attacks: either, the payer needs to prove knowledge of some part of the posted message in a way that depends on time, or the transaction center keeps the posted messages in a sorted list and ignores any repeated posting. If the latter approach is taken, the payment orders are encrypted using a new public key for each payment period (the time in between two accounting phases.)

The transaction center then may verify that the payer has sufficient funds. This may be required of certain banks, or for amounts above some threshold; alternatively, verifications may be performed randomly or in batches to lower the processing times and costs.

Each valid encrypted payment order \tilde{o}_i is added to an internal list of payments to be performed; there is one such list for each denomination. The payers' accounts are debited accordingly, either immediately or in batches. The transaction center signs each⁵ individual valid encrypted

⁵ Suites of transactions orders corresponding to the same transaction may be signed

payment order \tilde{o}_k , resulting in a signature σ_k . The transaction center uses different public keys for different denominations, or appends a description of the denomination before signing. It sends $\sigma_1, \dots, \sigma_n$ to the payer.

- (c) The payer sends a suite of signed, encrypted payment orders (of the format \tilde{o}_i, σ_i) to the corresponding merchant. He also sends a proof of what the encrypted messages are (in some cases, this amounts plainly to show the plaintext.)
- (d) The merchant verifies that σ_i is the transaction center's signature on the payment order \tilde{o}_i , and that the decrypted value o_i is of the form $m|a|s$ for valid a valid merchandise description m , the merchant's account number a , and a sequence number s not yet received.

If the above verification goes through, then the merchant stores the sequence number, and delivers the merchandise purchased.

3. Completion of Payment

At given intervals, the transaction center decrypts all the payment orders using the mix-decryption scheme employed. The result is a plaintext list of payment orders for each denomination. The items of these lists indicate what accounts are to be credited, and by how much. The banks corresponding to the accounts indicated credit these accounts accordingly.

Remark 1: Note that the payment orders can be generated by the payers without any real interaction between the payer and the merchant. This is possible if m is a publicly available description (e.g., the hash of an advertisement sent out by the merchant,) a is publicly available as well, and s_1, \dots, s_k is selected uniformly at random from a space that is large enough to avoid collisions.

Remark 2: In order to limit the amount of storage needed for the merchant, the sequence numbers may contain a date and time stamp, and will only be accepted - and stored - within an interval associated with this stamp.

5.3 Tracing

If a general mix-network is employed, only the two basic tracing operations can be performed efficiently, tracing from a payer to a payment order, and the opposite direction. We will later look at how this can be extended (and simplified) for an ElGamal based mix-network.

1. Payer \rightarrow Payment Order

The trace is performed simply by decrypting the encrypted payment order \tilde{o} in question, arriving at the plaintext payment order o , which specifies the account to which the payment is performed.

2. Payment Order \rightarrow Payer

The trace is performed as follows: the mix-servers reverse the computation

together to improve efficiency.

of o from \tilde{o} step by step, proving to each other that each step is valid. Depending on what type of mix-network is employed, it may be necessary to keep intermediary values, such as partial decryptions and the permutations employed; if these are not available, then they can be re-generated by decrypting the entire bulletin board again.

6 Paying and Tracing (ElGamal Version)

If the recently proposed type of mix-decryption scheme (different methods independently proposed by Ogata, Kurosawa, Sako, and Takatani [22], by Abe [1], and Jakobsson [19]) is used by the transaction center, this allows several improvements to be made. Let us first briefly describe this mix-decryption scheme, and then elaborate on the advantageous consequences of using it.

1. The input to the scheme is a list of ElGamal encrypted messages (encrypted with the public key of the transaction center, i.e., mix center.) The output is a permuted list of the corresponding decrypted values.
2. The decryption process can be performed by any k out of n servers, who share the secret key for decryption. The participants with whom the encryptions originated need not be involved.
3. No subset of less than k out of n mix servers can perform the decryption.
4. The decryption process is robust, meaning that if any server(s) should cheat, then the cheating will be detected, and their identities become known to the remaining servers.
5. No subset of cheating servers can correlate items of the input to items of the output (unless they already know the corresponding plaintext messages.)
6. The decryption process is efficient.

If this scheme is employed, the following holds:

1. As long as there exists a quorum of honest mix-servers (controlled by the banks), the decryption process will be possible. This is an important issue, since otherwise it would be possible for one corrupt bank (controlling one mix-server) to stop all payments in one payment period (unless the payers volunteer to repeat their payments!)
2. As long as at least one of the participating mix-servers is honest, the correctness of the output is guaranteed. (This is important, or a subset of mix-servers could manipulate the payments without being detected.)
3. As long as there is no dishonest quorum of mix-servers, the privacy of users will be guaranteed.

The tracing scheme can be simplified as well, and makes it unnecessary for the banks to store the contents of the bulletin board, or intermediary computation, in order to perform traces. The following three types of tracing can be performed:

1. **Payer** → **Payment Order**

The trace is performed simply by decrypting the encrypted payment order

\tilde{o} in question, arriving at the plaintext payment order o , which specifies the account to which the payment is performed.

2. **Payment Order \rightarrow Payer**

Here, the given decrypted payment order is to be compared to the encrypted input items. When a match is found, the corresponding payer identifier is output. This can be done by first blinding both the payment order and all the input items⁶ using the same distributively held blinding factor, and then decrypt all the blinded input items, after which a match is found. If the mix-decryption scheme by Jakobsson [19] is used, then the trace can be performed by computing a tag, corresponding to the input to the mix-network, and comparing this tag to the partially decrypted (but still blinded) encrypted payment orders. We refer to [19] for a more detailed description.

3. **(Payment Order, Payer) \rightarrow (yes/no)**

This trace is performed by computing the above tag from the encrypted payment order in question, and verify whether this tag corresponds to the decrypted payment order as $tag = o^{1/\delta}$. This can be done using the verification protocol for undeniable signatures [7], or using a small modification of the first method described above.

7 Claims

The basic scheme satisfies the following requirements (for a quorum size of all participating bank servers, who are assumed to follow the protocol, although they may be curious to learn additional information): *unforgeability*, *impersonation safety*, *overspending blocking*, *payment blocking / revocability*, *uniform anonymity*, and *framing-freeness*. Additionally, the two first modes of tracing detailed in *traceability* are possible.

The ElGamal based scheme satisfies (without any condition on quorum size or behavior of non-cooperating servers) the above requirements, and full *traceability*. We prove these claims in the Appendix.

8 Performance Analysis

We have considered two different approaches when implementing the scheme:

- mobile approach: the user’s device is a portable token, such as a smart-card with restricted capacities in terms of computational power and memory storage;
- PC-based approach: in this case, the user could rely on some added resources like a desktop or lap-top computer; such a configuration opens the door to various improvements such as performing server-aided computation or subcontracting public-information related processing.

⁶ An ElGamal encryption (a, b) can be blinded using a blinding factor δ by computing (a^δ, b^δ) . This results in a blinding of the plaintext message m to m^δ , but in encrypted form.

8.1 General Figures

Table 1 summarizes computational performance of two smartcards with cryptographic accelerators: the new smartcard product from Siemens [24], used for instance in the Gemplus GPK range, and the recent and promising Hitachi[14]⁷ enhanced chip for smartcards. Performance of the Siemens SLE66X160S has been carefully evaluated, running routines on emulator at a 3.68 MHz clock frequency and with the worst-case set of parameters (all the exponent bits set).

Data Length l (bits)	512	768	1024
H8/3113	90 ms	350 ms	700 ms
SLE66CX160S	186 ms	593 ms	1045 ms

Table 1. Smarcard - Exponentiation Timings for exponent= l

PC level of performance depends both on hardware (processor cache size and clock frequency, type of memory, etc.) and implementation techniques (number representation, inline assembly, custom fixed-size implementation). We deal mainly with some specific techniques in order to provide a good level of performance, namely:

- Montgomery’s Method: speed-up modular multiplication up to 20
- Inline Assembly code: custom implementation with fixed-size modulus (very fast but at the cost of code expansion);
- t -bit window: define a sliding t -bit window to process groups of t bits of the exponent; $t = 5$ is optimal for popular modulus sizes ranging from 512 to 1024 bits.
- True 32-bit executables: compiling source codes with a 32-bit compiler eventually improves performance significantly.

Table 2 gives an overview of these different techniques and the related performance. A very basic 60-Mhz Pentium was used as a testing machine; we add successively the different improvements for a precise evaluation of the impact on performance, i.e second line of the table means that we combine montgomery and inline assembly techniques.

8.2 Implementation Performance

We implemented both scenarii, token-based only implementation and PC/smartcard combination. The smartcard used is based on the Siemens chip (GPK card)

⁷ Timings are estimated since the chip will be introduced in 1998.

Data Length l (bits)	256	512	1024
Montgomery's Method	38 ms	201 ms	1345 ms
Inline Assembly Code	21 ms	102 ms	673 ms
5-bit window	20 ms	97 ms	645 ms
True 32-bit	≤ 15 ms	70 ms	407 ms

Table 2. PC - Exponentiation Timings for balanced exponents

giving us timings around 300 ms for the 1024-bit RSA signature (with CRT) and 200 ms for the El-Gamal encryption. The total time expected for processing a payment order is approximately the time required for encrypting and signing data: The communication time for processing such transactions is around 100 ms when transmitted according to 7816-4 standard at 115,200 bd. Since the user does not need to update a transaction log, writing to EEPROM (which is quite time consuming) is avoided.

Smartcard Performance: We performed various practical tests confirming that the total time for issuing a payment order should be around 1 second, as detailed in table 3. We assume that:

1. the time spent for checking at merchant and center is negligible (we consider a protocol overhead of 100 ms)
2. communication is performed at 115,200 bds
3. 512-bit modulus for ElGamal and 1024-bit modulus for RSA are practical and secure parameter sizes

PC/Smartcard Implementation: Using a PC is particularly suitable when many payment orders are processed. The idea is that we can subcontract the bulk encryption and possibly part of the hashing, depending on the context, smartcard performing the signature plus minimal computation for control of the transaction. In this case, where multiple payment requests are signed using the same signature, the average cost of a transaction is reduced, shifting memory and computational requirements from the user's token to the local PC.

A final remark: considering Hitachi's level of performance, a specialized circuit for modular multiplication could achieve 1024-bit RSA signature in less than 200 ms.

9 Acknowledgements

We want to thank Moti Yung and David Kravitz for interesting discussions before resp. during the conception and writing of the article.

Operation	Processing Time
512-bit El-Gamal Encryption	227 ms
Hashing Ciphertext (SHA)	32 ms
1024-bit RSA Signature	289 ms
Sending to center	110 ms
Sending to merchant	110 ms
Checking Overhead	100 ms
Total	868 ms

Table 3. Transaction Time Evaluation

References

1. M. Abe, "Universally Verifiable Mix-net with Verification Work Independent of the Number of Mix-centers," Eurocrypt '98.
2. E. Brickell, P. Gemmell, D. Kravitz, "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change," Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1995, pp. 457-466.
3. J. Camenisch, U. Maurer, M. Stadler, "Digital Payment Systems with Passive Anonymity-Revoking Trustees," Computer Security - ESORICS 96, volume 1146, pp. 33-43.
4. J. Camenisch, J-M. Piveteau, M. Stadler, "An Efficient Fair Payment System," 3rd ACM Conf. on Comp. and Comm. Security, 1996, pp. 88-94.
5. D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, ACM 1981, pp. 84-88
6. D. Chaum, "Blind Signatures for Untraceable Payments," Advances in Cryptology - Proceedings of Crypto '82, 1983, pp. 199-203.
7. D. Chaum, H. Van Antwerpen, "Undeniable Signatures," Crypto '89, pp. 212-216
8. D. Chaum, A. Fiat and M. Naor, "Untraceable Electronic Cash," Advances in Cryptology - Proceedings of Crypto '88, pp. 319-327.
9. G.I. Davida, Y. Frankel, Y. Tsiounis, and M. Yung, "Anonymity Control in E-Cash Systems," Financial Cryptography '97.
10. Y. Frankel, Y. Tsiounis, and M. Yung, "Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-Cash," Advances in Cryptology - Proceedings of Asiacrypt 96, pp. 286-300.
11. M. Franklin and M. Yung, "Towards Provably Secure Efficient Electronic Cash," Columbia Univ. Dept of C.S. TR CUCS-018-92, April 24, 1992. (Also in Icalp-93, July 93, Lund Sweden, LNCS Springer Verlag).
12. M. Franklin and M. Yung, "Blind Weak Signatures and its Applications: Putting Non-Cryptographic Secure Computation to Work," Advances in Cryptology - Proceedings of Eurocrypt '94
13. Gemplus, "Gemplus Public Key (GPK) Cards", October 1996.
14. Hitachi, "Product directory: H8 Range", 1997.
15. M. Jakobsson and M. Yung, "Revocable and Versatile Electronic Money," 3rd ACM Conference on Comp. and Comm. Security, 1996, pp. 76-87.
16. M. Jakobsson and M. Yung, "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System," Financial Cryptography '97.

17. M. Jakobsson and M. Yung, "Distributed 'Magic Ink' Signatures," Eurocrypt '97, pp. 450-464
18. M. Jakobsson, "Privacy vs. Authenticity," PhD Thesis, University of California, San Diego, Department of Computer Science and Engineering, 1997. Available at <http://www-cse.ucsd.edu/users/markus/>.
19. M. Jakobsson, "A Practical Mix," Eurocrypt '98, available at www.bell-labs.com/user/markusj/
20. D. M'Raihi, "Cost-Effective Payment Schemes with Privacy Regulation," Advances in Cryptology - Proceedings of Asiacrypt '96.
21. National Institute for Standards and Technology, "Digital Signature Standard (DSS)," Federal Register Vol 56(169), Aug 30, 1991.
22. W. Ogata, K. Kurosawa, K. Sako, K. Takatani, "Fault Tolerant Anonymous Channel," Information and Communications Security, '97, pp. 440 - 444
23. R. Rivest, A. Shamir and L. Adleman, "A method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, v. 21, n. 2, Feb 1978, pp. 120-126.
24. Siemens, "ICs for Chip Cards", Customer Information, June 1997.
25. S. von Solms and D. Naccache, "On Blind Signatures and Perfect Crimes," Computers and Security, 11 (1992) pp. 581-583.
26. M. Stadler, J-M. Piveteau, J. Camenisch, "Fair Blind Signatures," Advances in Cryptology - Proceedings of Eurocrypt '95, 1995.
27. P. Syverson, D. Goldschlag, M. Reed, "Anonymous connections and onion routing," IEEE Symposium on Security and Privacy, 1997.
28. B. Witter, "The Dark Side of Digital Cash," Legal Times, January 30, 1995

Appendix: Proofs of Claims

We prove the claims stated in section 7; where applicable, the proofs have one part relating to the general scheme (assuming that all mix servers of the mix-network used cooperate) and one to the ElGamal scheme (assuming a quorum of specified size to cooperate).

Theorem 1: The schemes implement unforgeability.

Proof of Theorem 1: (*Sketch*)

First, in order to post a payment order that will be accepted, a valid signature of the payer has to be produced; by the soundness of the underlying existentially unforgeable signature scheme, this can only be done by the account owner in question. Second, in order for a party to be credited, it is necessary that one of the the output items of the mix-network specifies their account number. If a general mix-network is used, the mix-servers are assumed to perform the valid decryption (if they do not, they can only alter who gets credited, and not the number of payments performed); if the suggested ElGamal mix-network is used, then by the robustness of this, the outputs are guaranteed to be valid if at least one participating mix server is honest. Therefore, only the participating payers will be charged, and only the intended payees debited. The amount of the credits cannot exceed the amount of debits. \square

Theorem 2: The schemes implement impersonation safety.

Proof of Theorem 2:

By the soundness requirements of the identification schemes used to obtain access to accounts, only authorized parties are going to get access to their accounts, as long as they keep their secret keys secret. Also, it is not possible for an adversary to produce a signature by a payer not cooperating with him. This follows from the assumption that the signature scheme used by the payer to sign an encrypted payment order is existentially unforgeable: a new signature cannot be produced without knowledge of the secret key. If the same signed message is reposted in the same payment period, then it will be removed, either after the payer has failed to prove knowledge of some part of the post, or since duplicates are ignored (depending on the approach taken.) Since we, for the approach where duplicates are removed, require the payment orders to be encrypted using a public key specific to the time period between two accounting sessions (corresponding to the mix-decryptions), and the relationship between the secret keys of intervals is unknown, it is not possible to force an old encryption to be accepted in a new time interval.

Theorem 3: The schemes implement overspending blocking.

This follows trivially from Theorem 1, and the fact that for each payment one signature has to be generated, and for each such signature, one account is billed.

Theorem 4: The schemes implement payment blocking / revocability.

Proof of Theorem 4: (Sketch)

Since a payment can only be made from an account by producing a signature, and sending this to the transaction center, and the signatures used for this will identify the account owners, it is possible for an account owner to stop all payments from his account by requesting that no signature of his is accepted by the transaction center. This corresponds to either putting his public key on a blacklist, or by removing it from the list of valid public keys. Similarly, the same can be done by the bank of an account holder to block the account holder access to his funds. □

Theorem 5: The schemes implement framing-freeness.

Proof of Theorem 5: (Sketch)

This follows from Theorem 2, and the fact that nobody but a user (including the bank servers) knows the secret key of this user. Therefore, it is not possible to produce a set of transcripts indicating that a party performed a given payment, without the cooperation from this party. □

Theorem 6: The schemes implement uniform anonymity.

Proof of Theorem 6: (Sketch)

The only way a payment can be initiated is by posting an encrypted and signed payment order on the bulletin board of the transaction center. The signature identifies the payer; if the identity of the signer cannot be established (or he is not authorized to make payments,) then the posted message will be ignored. Then, the only way that merchants can be credited is by decrypting all the

payment orders. The link between encrypted and decrypted items can only be established by a quorum of mix-servers. If fewer than this could correlate the input and output items, this contradicts the assumption that the ElGamal encryption scheme is probabilistic. By the traceability option (see next theorem), the anonymity of any valid encrypted or decrypted payment order can be removed, and therefore, all participants enjoy the same degree of anonymity. \square

Theorem 7: The general scheme implements the two first methods for traceability; the ElGamal based scheme implements full traceability.

Proof of Theorem 7: (*Sketch*)

We have established above that for each account that is credited, the transaction center has a signature of the party whose account will be debited: the link between the two can always be established by either decrypting a single posted encrypted and signed payment order (tracing from payer to payment order), or by (potentially partially) re-encrypting a decrypted payment order (arriving at the payer information from the payment order.) In addition, the third tracing option (comparison) can be performed in the ElGamal based scheme, by the use of the verification protocol for undeniable signatures. \square