International Journal of Computer and Communication Technology

Volume 5 | Issue 3 Article 8

July 2014

Codex Enables Secure Offline Micropayments

A B. Sagar
DCIS University of Hyderabad Hyderabad, bablusagar@gmail.com

Codex Enables Secure Offline Micropayments

A B Sagar
Department of Computers and Information Sciences
Hyderabad Central University
Gachibowli, Hyderabad
Email: bablusagar@gmail.com

***Abstract*—**This paper introduces a new micropayment scheme, suitable for all kinds of transactions, and does not require online transactions for either the payer or payee. The designed method uses an encrypted data structure called Codex which self replicates to represent the current values of both the payer and the payee. The model, while providing fraud detection also guarantees payment & loss recovery.

Keywords:*offline micropayments, codex, secure micropayments*

I. PROBLEM DESCRIPTION

Current offline payment systems are not well matched to occasional, low-valued transactions. (For the purposes of this discussion, we use the term “electronic payment system" broadly, to encompass conventional credit cards, debit cards, online and offline digital cash, etc.) Online electronic payment systems (which require access to a server for each transaction) provide security but they cause overload for the issuing authorities such as “Banks”. Also, as these are micropayments, and their currency value is relatively lower, the overload for the banks to supervise each and every micropayment makes it unnecessary burden. And for the users, since micropayments are done very frequently, contacting bank for each and every transaction is an overload and needs to be avoided. In addition, considering the online connectivity issues, we can infer that an offline payment system (which allows transactions with no server) is very important. Currently the computational capacities of all hand held devices has increased drastically, so a computation intensive but secure payment system is agreeable to most of the users. Electronic payments face several challenges such as double spending, counterfeiting, man in the middle, etc. Loss recovery, guarantee of payment and fraud detection are other features that are much required in an electronic payment system. Till now several electronic payment systems were proposed but there is not yet a sustainable offline electronic payment system which is not susceptible to double spending, counterfeiting, man in the middle attack, etc and has features like fraud detection, loss recovery and guarantee of payment. Our objective is to develop a framework for such an offline micropayment system which has features like sustainability, fraud detection, loss recovery and guarantee of payment while resistant to double spending, counterfeiting and man in the middle attack.

II. RELATED WORK

Most currently-used protocols for Internet e-commerce are based on credit card charging over SSL [Hic95]. Such schemes require the merchant to perform a hidden (from the user’s point of view) online credit check. The cost of such checks can be on the order of 10 (US) cents, making them expensive for low-value transactions. The more recently developed SET [SET] and CyberCash protocols [EBCY96] do not address this issue.

NetBill [CTS95] is a transactional payment protocol with many advanced features (atomicity, group membership, pseudonyms, etc.) that require communication with the NetBill server for each transaction, thus exhibiting the same drawback with respect to micropayments as the simpler online protocols already mentioned. Other general-purpose payment protocols [NM95, BGH95, FB98] are unattractive for micropayments for these same reasons.

Digital cash-based systems [Cha82, Cha92, MN94, BGJY98, dST98] provide many desirable features (potentially total anonymity, inherent off-line operation), but do not directly address the issue of double-spending (fraud). Some e-cash systems use online checking (thus negating the off-line operation capability). Others rely on detection after the fact, which introduces the potential for large-scale simultaneous multiple-spending. The same drawback is manifest in several micropayment procotols, such as PayWord [RS], PayTree [JY96], micro-iKP [HSW96], and others [Tan95]. While the double-spending possibility is an inherent property of all such systems, none of the above protocols employ any kind of risk management scheme to address it.

NetCents [PHS98] and Millicent [Man95] are scrip-based off-line-friendly micropayment protocols. As the monetary unit used in these protocols is vendor-specific, double-spending is made very difficult (if not impossible). The assumption behind both protocols is that people tend to re-use the same merchants repeatedly. If this assumption holds, the interactions between the customer and the bank are kept at a minimum. A hidden assumption is that merchants have ¸Stotal information ˇT over their sales, so double-spending with the same merchant is detectable. If the merchant has many distinct points of sale, the potential for double-spending is re-introduced, unless continuous communication and database synchronization is maintained between the different points. This would consequently negate the benefits of off-line operation.

IBM’s MiniPay [HY96, Her98] uses a protocol that is somewhat similar to that described in this paper. MiniPay was developed primarily for use within a web browser, and a lot of effort has gone into the user interface aspect. Risk management is implemented as a decision to perform an online check with the billing server based on the total spending by the customer that day, and some parameter set by the merchant. The billing provider cannot customize risk-management parameters on a per-customer and/or per-merchant basis.

Person-to-person (P2P) payment systems, such as PayPal or X.com (now merged), allow users to exchange money online. Typically, the provider’s web server needs to be contacted and an instruction issued for a money transfer. In that respect, the transaction is very similar to a bank wire transfer. There also exist modules that allow users to directly exchange money through palmtop computers. Such systems typically have no built-in security mechanisms; in the best of circumstances, they are a straight variant of offline digital cash.

While our system can operate on its own, it could also be integrated in some type of electronic wallet. Finally, the use of a PDA as an electronic wallet is not new and our system can be built into a PDA.

III. DESIGN & IMPLEMENTATION

We propose a new micropayment scheme, suitable for micropayments, and does not require online transactions for either the payer or payee.

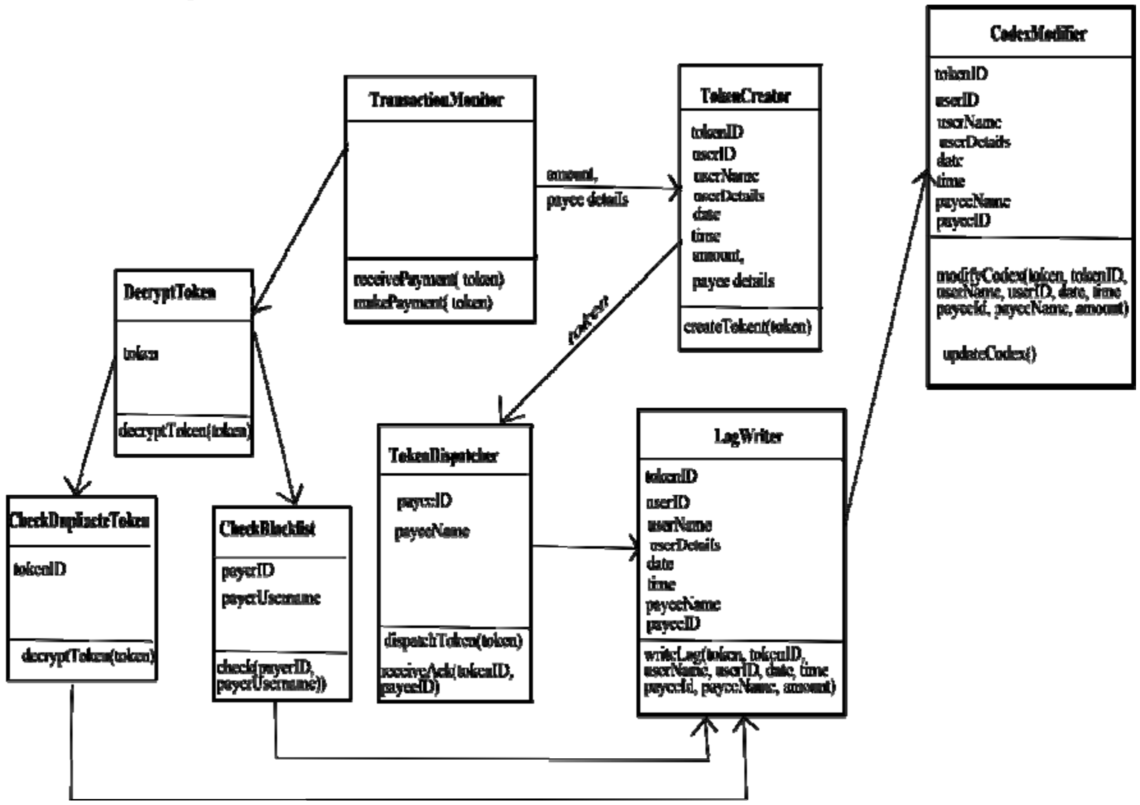


Fig. 1: Class Diagram of User

Some important classes used in the implementation of the model.

TransactionMonitor: It is a continuously monitoring process on the user end and it specifies the respective execution path for the process. It receives two commands, 'receive payment' and 'make payment'. If the command is make payment, then it follows the path: 'create token' → 'send to payee' → 'Write Log' → 'Modify and Update Codex'.

If the command is receive payment, then it follows the path: 'receive & decrypt token' → 'check blacklist' → 'Write Log' → 'Modify and Update Codex'.

Create Token: This process creates a payment token. The token identifies the payer and the payee. The token also has a token which identifies it uniquely.

TokenDispatcher: This process sends the token to the payee and also ensures that it is received properly. It waits until it receives the acknowledgment from the payee.

CodexModifier: This process modifies the Codex of both the payer and the payee to represent the present value, contain the token id, and the present transaction.

Receive & Decrypt Token: Every payment involves a token. Since the token contains information which is necessary for detecting and avoiding fraud, it is usually encrypted. This token is decrypted in this process and values are extracted.

Check Blacklist & Send Ack : The bank creates and maintains a database of fraudulent users called the blacklist. The user also gets a copy of the blacklist when he becomes a member, and also he can update his blacklist whenever he feels the need. In the decrypted token one of the value that is extracted is the payer. Now this payer is checked against the blacklist; if the user is not found in the blacklist, token is accepted. An Ack message is also sent to the payer acknowledging the receipt and acceptance of token.

LogWriter: Information regarding the token id, token, date, time, transaction, etc are written to log. This log will be later copied down by the bank. The bank uses this log to balance all the transactions and updates its own database. Thus the bank has a fair idea on how much a user has in his account, though he/she has not approached the bank. For example, if Alice becomes a user of this micropayment scheme with an initial amount of Rs. 10 and so does Bob with Rs 10. After creating the accounts of Alice and Bob, the bank also maintains its own database of its users along with their account information. If Alice makes payments worth Rs 10 to Bob, then her account has zero balance. She then goes to the bank to credit her account with more money, the bank before crediting amount into her account (i.e. updating her Codex), they copy her previous Codex and extract all her transactions. Through her transactions, it is evident that she paid Rs 10 to Bob. So they add this information to Bob's account. Thus the bank knows that Bob has Rs 600 in his account, though Bob has not been to the bank yet. Later Bob comes to the bank to credit money to his account or withdraw cash out of his account, and since the bank already knows how much Bob has in his account it can detect if Bob's account is prone to fraud or not.

- **ActivityMonitor**: It is a continuously monitoring process on the bank end and it specifies the respective paths for the processes. It receives five commands, 'create account', 'delete account', 'update Codex', 'create blacklist', and 'check consistency'. Depending on the command, it decides the execution path for the process.

- **AccountCreator**: The process creates an account for a new user.

- **CodexCreator**: The process creates a new Codex encrypting the username, userid, amount, expiry date, and other essential information.

- **UpdateBankRepository**: The process updates the bank's database with the new user's information and Codex details.

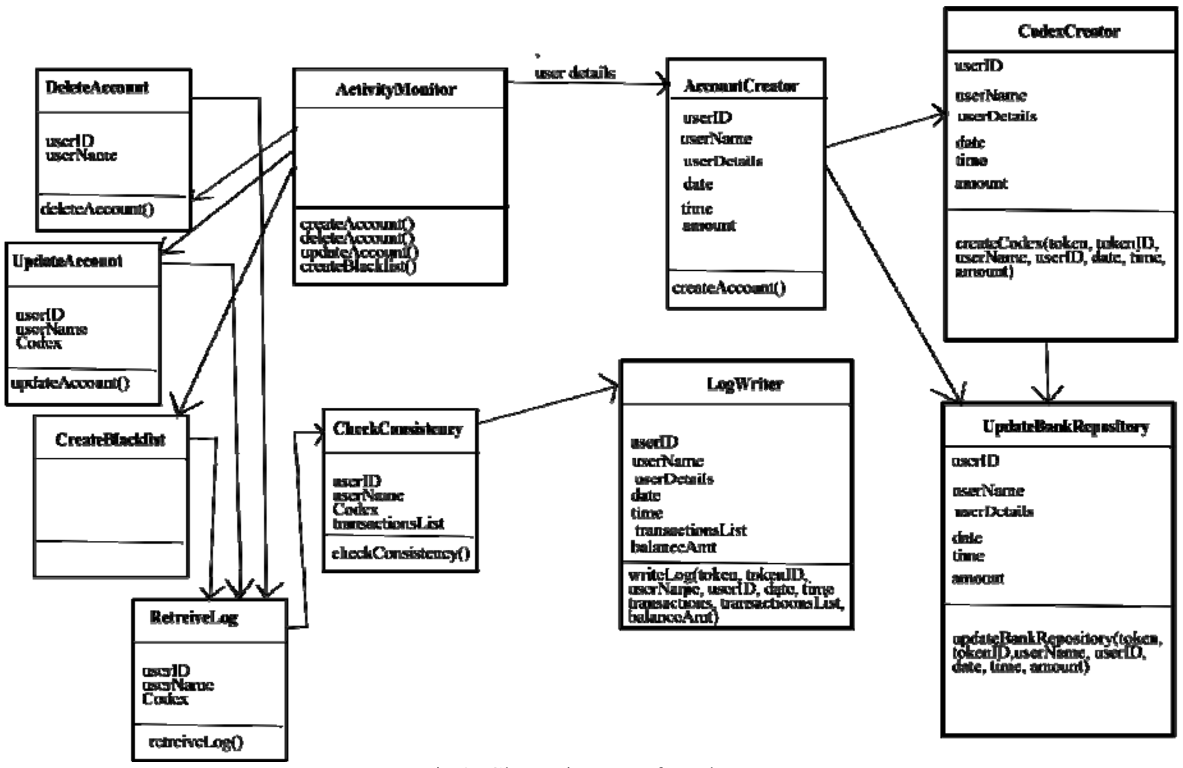


Fig 2: Class Diagram of Bank

• **RetrieveLog:** Before updating an account (i.e. adding credit balance to an account), deleting an account or creating a blacklist, the log contained in the user's Codex is retrieved.

• **Check Consistency:** The process checks for the consistency of the transactions by balancing them. However, the total amount in all the user's accounts should be equal to the total amount dispersed by the bank. If this condition is not satisfied then it indicates the possibility of fraud. Also, as the Token_Ids of each transaction are saved, tracing the fraudulent transactions becomes easier.

• **UpdateAccount:** If the transactions are found to be balancing, then the process updates the Codex of the user to represent the new amount.

• **DeleteAccount:** If the transactions are found to be balancing, then the process safely deletes the account.

• **CreateBlacklist:** If the transactions are not balancing, then the users whose accounts show wrong data are identified and added to blacklist.

IV. STRUCTURE OF CODEX

Codex, which is in an encrypted form, is comprised of the following four layers.

User Information Layer: This layer consists of user's details like username, userid, address, account creation date, etc. This layer is basically for user identification and user information.

Financial Information Layer: This layer consists of financial information of the account, details such as present balance, secure deposit at the bank, number of transactions that were made, list of all transactions, token IDs that were generated, token IDs that were received, etc.

Management Layer: This layer consists of information required by the management. Information such as when the account is going to be expired, Log of the user, performance Index of the user, etc.

Security Layer: This layer is meant for proactive fraud detection and prevention. Security layer consists of a blacklist so that before a transaction could occur, the user's process checks whether the payer is a fraudster or not.

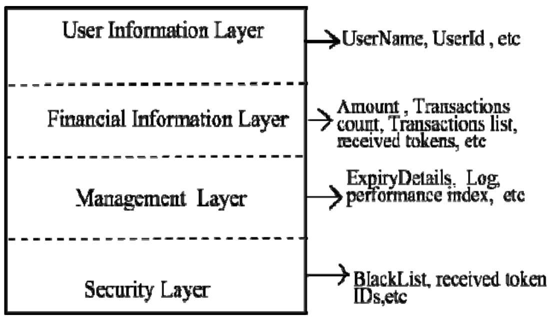
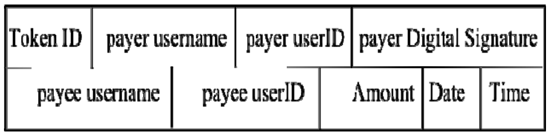

User Information Layer
Financial Information Layer
Management Layer
Security Layer
UserName, UserId , etc
Amount , Transactions count, Transactions list, received tokens, etc
ExpiryDetails, Log, performance index, etc
BlackList, received token IDs,etc

Fig 3: Structure of Codex



Token ID	payer username	payer userID	payer Digital Signature	
payee username	payee userID	Amount	Date	Time

Fig. 4: Structure of Payment Token

V. OPERATIONAL BEHAVIOR OF THE PROPOSED MODEL

The proposed model functions using an encrypted message called Codex, and it assumes that the algorithm used for generation of Codex is protected from physical theft. Codex is a dynamically updated encrypted message and contains user's details such as user_name, user_id, amount, count of transactions, list of all previous transactions specifying transaction_id, date, amount, payee_name, payee_id, etc for each transaction. Also each codex is uniquely identifiable.

When a user Alice registers as a member of the payment scheme, the bank will create an account for her and a Codex is initialized for Alice. This Codex is also saved in bank's database against Alice's account. Suppose, Alice wants to pay an amount X to Bob, then two things happen on Alice's side. Firstly, a payment token which contains the details such as token id, amount, date, time, transaction id, payer username, payer id, payee username, payee id,etc in encrypted form is generated from Alice's Codex.

Secondly, the Codex of Alice is updated. The Codex modified so as to represent the present balance, count of transactions, and also the token id and token. The more payments Alice makes, the more payment tokens are generated, and Codex is updated to contain all the transactions details.

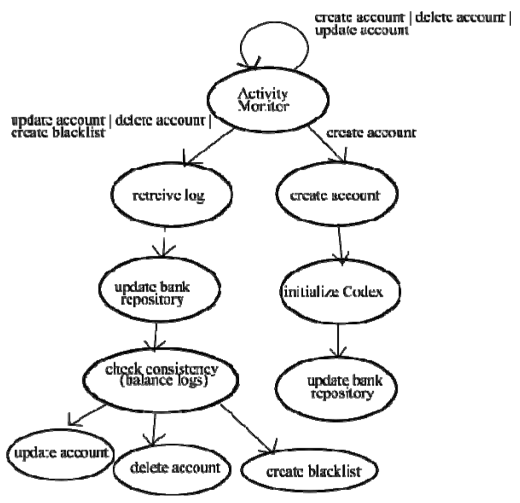


Fig 5: Architecture

On Bob's side, after receiving the payment token from Alice, Bob updates his Codex such that it now represents his new balance, and contains the present transaction details, payment token id.

A. *Axioms:*

$*\ \forall\ T_i, T_j \in \{T \mid T \text{ is a Token}\}, T_i.\text{tokenID} \neq T_j.\text{tokenID}$

$*\ \forall\ T_i \in \text{token}, s_1, s_2 \in \text{Time} \mid s_1 \neq s_2, \text{generated}(T_i,s_1) \neq \text{generated}(T_i,s_2)$

$*\ \text{pay}(X,Y) \Rightarrow \text{createToken}(X) \wedge \text{updateCodex}(X) \wedge \text{updateCodex}(Y)$

$*\ \text{createToken}(X) \Rightarrow \text{Amount}(X) > 0 \wedge (\text{value(Token)} < \text{current\_balance}(X)) \wedge \neg \text{expired}(X)$

$*\ \text{updateCodex}(X) \Rightarrow \text{update(Codex, new Amount)} \wedge \text{add(Codex,TokenId)} \wedge \text{add(Codex, trans\_details)}$

VI. SPECIAL FEATURES OF THE SYSTEM

The proposed model has some special features be-sides allowing payments offline.

A. Fraud Detection

There are several possibilities of fraud since the payment is not in physical form but in digital form. Even physical form of money also has problem of fraud as the fraudsters can print fake currency notes. However, we can solve the problem of fraud in Ecash to a maximum extent using cryptographic techniques and a consistent model that supports fraud detection. Our model makes room for fraud detection and prevention.

Proactive Fraud Detection: Detecting the fraud after it has happened is not always beneficial. Our model supports proactive fraud detection and provides two techniques for prevention of fraud. First technique is that every user is provided with a blacklist of fraudsters. And, before a user receives a payment from any payer, the user checks to see whether the payer is in the blacklist. If the payer's id is found in the blacklist, the transaction is canceled. Secondly technique is useful when the payer's name is not in the blacklist. This is the case when the payer might try to generate payment tokens though he does not have amount in his account. Our model overcomes this problem because of the Codex. The Codex is encrypted and self modifies to reflect the current balance of the account. When the token generating algorithm accesses the Codex, it realizes that the amount is zero, and token generating activity is canceled.

Double Spending: The problem of double spending comes only with Ecash. There are two cases in double spending. Firstly, a payer might try to pay the same payment token to two people. To overcome this problem, our model insists on generating tokens which are specific to a payee. Thus, if a user 'A' wants to generate a token 'Ti' for a payee 'B', then 'B' is required to provide his username and userid. Then, A will generate a payment token that can only be accepted by 'B'. On B's end, before receiving any payment token, B's process will check the payment token whether it is meant for B or not. In this manner, double spending can be avoided.

Secondly, a payer may try to pay with the same token to the same payee. So, in this case checking for specificity alone then it is not enough. It brings us to the need of identifying each token. Hence the token generating algorithm includes a 'token ID' in each payment token. Now, if the payee receives the same 'token ID' twice, then he will be able to identify and reject it. Also, each token has date and time stamps which help in hinting if the token is an old one.Man-in-the-middle: Man in the middle problem is a situation where someone is eavesdropping. The eavesdropper tries to capture the payment tokens and tries to pay to the payee as though they are his own tokens. Our model overcomes this problem by making the payment token carry the digital signature of the payer. So, even if someone does eavesdropping and stole the token, they cannot claim it.

B. Guarantee of Payment & Loss Recovery:

The model provides several measures to overcome fraud. Proactive fraud detection and prevention methods prevent most of the fraudulent payments. But still if the mechanism did not work properly and it was identified that a user was made a false payment, then the Bank identifies the payer by balancing its accounts. It may take time for the bank to balance the accounts because all the users will not approach the bank at the same time. To deal with this issue, the bank may so design the Codex that it will expire after a limited period of time or after a specified number of transactions. In such a case, the fraudster cannot continue making false payments forever. The bank also makes the users make a safe deposit for each account. And, when a false payment was identified, and the payer was also identified, the bank will make the payment from the safe deposit of the false payer. Thus the user is guaranteed of payment. The bank also recovers its loss from the safe deposit. Despite all the security measures implemented by the bank, if still some false payments are made, the loss will be minimal since these are micropayments.

VI. DISCUSSION AND CONCLUSION

We have demonstrated a simple and, for some applications, practical scheme for offline micropayments without the overhead of online transaction authorization. Our scheme represents a departure from the usual approach to designing such systems. In particular, we chose to prevent losses, rather than allowing them by providing anonymity.

VII. FUTURE WORK

- Developing the Codex and encryption for the Codex
- Developing the Payment Token
- Developing the algorithms that modify the Codex, receive payment tokens, etc

ACKNOWLEDGMENTS

The author likes to thank Prof Hrushikesha Mohanty for his suggestions and research advices explored in this paper.

REFERENCES

[BFIK99] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System Version 2. Internet RFC 2704, September 1999.

[BFL96] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In Proc. of the 17th Sympo- sium on Security and Privacy, pages 164 -173. IEEE Computer Society Press, Los Alamitos, 1996.

[BGH95] M. Bellare, J. Garay, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP - A Family of Secure Electronic Payment Protocols. In Proceedings of the First USENIX Workshop on Electronic Commerce. USENIX, July 1995.

[BGJY98] M. Bellare, J. Garay, C. Jutla, and M. Yung. VarietyCash: a Multi-Purpose Electronic Payment System. In Proceedings of the Third USENIX Workshop on Electronic Commerce. USENIX, September 1998.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In Advances in Cryptology: Crypto ’82 Pro- ceedings. Plenum Press, 1982.

[Cha92] D. Chaum. Achieving Electronic Privacy. Scientific American, pages 96 -101, August 1992.

[CTS95] B. Cox, D. Tygar, and M. Sirbu. NetBill security and transaction protocol. In Proceedings of the First USENIX Workshop on Electronic commerce. USENIX, July 1995.

[DB99] N. Daswani and D. Boneh. Experimenting with Electronic Commerce on the PalmPilot. In Proceedings of the Third International Conference on Financial Cryptography, number 1648 in Lecture Notes in Computer Science, pages 1 -16. Springer-Verlag, 1999.

[DBGM98] N. Daswani, D. Boneh, H. Garcia-Molina, S. Ketchpel, and A. Paepcke. SWAPEROO: A Simple Wallet Architecture for Payments, Exchanges, Refunds, and Other Operations. In Proceedings of the Third USENIX Workshop on Electronic Commerce. USENIX, September 1998.

[dST98] A. de Solages and J. Traore. An Efficient Fair Off-Line Electronic Cash System with Extensions to Checks and Wallets with Observers. In Proceedings of the Second International Conference on Financial Cryptography, number 1465 in Lecture Notes in Computer Science, pages 275 -295. Springer- Verlag, 1998.

[EBCY96] D. Eastlake, B. Boesch, S. Crocker, and M. Yesil. CyberCash Credit Card Protocol Version 0.8. Internet RFC 1898, February 1996.

[FB98] E. Foo and C. Boyd. A Payment Scheme Using Vouchers. In Proceedings of the Second International Conference on Financial Cryptography, number 1465 in Lecture Notes in Computer Science, pages 103 -121. Springer-Verlag, 1998.

[Her98] A. Herzberg. Safeguarding Digital Library Contents. D-Lib Magazine, January 1998.

[Hic95] K. Hickman. Secure Socket Library. http://home.netscape.com/security/techbriefs/ssl.html February 1995.

[HSW96] R. Hauser, M. Steiner, and M. Waidner. Micro-payments based on ikp. In Proceedings of the 14th Worldwide Congress on Computer and Communication Security Protection, June 1996.

[HY96] A. Herzberg and H. Yochai. Mini-Pay: Charging per Click on the Web. http://www.hrl.il.ibm.com/mpay/, 1996.

[JY96] C. Jutla and M Yung. Paytree: amortized signature for flexible micropayments. In Proceedings of the Second USENIX Workshop on Electronic Commerce. USENIX, 1996.

[Man95] M. S. Manasse. The Millicent protocols for electronic commerce. In Proceedings of the First USENIX Workshop on Electronic Commerce. USENIX, July 1995.

[MN94] G. Medvinsky and C. Neuman. NetCash: A design for practical electronic currency on the internet. In Proceedings of the Second ACM Conference on Computer and Communication Security, November 1994.

[NM95] C. Neuman and G. Medvinsky. Requirements for network payment: The Netcheque prospective. In Proceedings of IEEE COMCON, March 1995.

[PHS98] T. Poutanen, H. Hinton, and M. Stumm. NetCents: A Lightweight Protocol for Secure Micropayments. In Proceedings of the Third USENIX Workshop on Electronic Commerce. USENIX, September 1998.

[RS] R. Rivest and A. Shamir. PayWord and MicroMint. CryptoBytes, 2(1):7 -11.

❑❑❑