Preliminary Report on Chaum's Online E-Cash Architecture

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

February 28, 2008

Abstract

This report discusses the security of Chaum's online e-cash architecture [2]. This architecture can be used with an arbitrary blind signature scheme to give an anonymous e-cash system. This should be regarded as a preliminary report after a short period of research.

1 Introduction

In 1982, Chaum introduced the concept of an anonymous e-cash scheme [2]. The e-cash system that he proposes can be separated into two distinct parts: an online e-cash architecture and a blind signature scheme. The online e-cash architecture can be combined with an arbitrary blind signature scheme to create an anonymous e-cash scheme. Blind signature schemes have received a large amount of study and are out of scope of this report. This report will concentrate on the security properties of the online architecture.

For the purposes of this report, an e-cash scheme is a set protocols between a series of entities:

- The bank. The bank consists of several distinct parts: a central authority which sets up the system; a mint that issues coins; a deposit counter where coins can be redeemed; and a doublespend database (DSDB) which ensures that coins aren't spent twice.
- The user. The user exchanges money for coins with the bank (mint) and spends them with the merchant.
- The merchant. The merchant receives coins from the user, checks their authenticity with the bank (the deposit counter and the doublespend database), and exchanges them for money.

The main processes within an e-cash scheme are:

- `Setup`: The protocol executed by the bank (central authority) to set up the entire e-cash system.
- `User registration`: If necessary, the user registers their identity with the bank. This may or may not involve the user generating a public/private key pair for their identity.
- `Merchant registration`: If necessary, the user registers their identity with the bank. This may or may not involve the merchant generating a public/private key pair for their identity.
- `Withdraw`: A protocol executed between the user and the bank (mint). The user somehow pays a set amount of money and is given a coin of a particular denomination. This key is typically identified by a serial number that is chosen at random by the user. It will become important that the bank does not gain any information about a coin's serial number during this process.

- `Spend`: A protocol between the user and the merchant. The user provides the merchant with the coin and some assurances that the coin is valid for a particular denomination – i.e. that it has been produced by a valid mint.
- `Deposit`: A protocol between the merchant and the bank (deposit counter and doublespend database). The merchant presents the coin to the bank along with assurances that it is valid. This is checked by the deposit counter and the serial number of the coin is extracted. The serial number is compared to the doublespend database to ensure that each coin is only spent once. The merchant is then given a set amount of money in return for the coin.

Individual schemes may have more functionality then is present in the basic model here. We also note that many schemes collapse the `Spend` and `Deposit` protocols into a single protocol between the user, merchant and bank with an inline structure (in which both the user and the bank communicate with the merchant, but do not communicate with each other). This combined `Spend-Deposit` protocol checks the validity of the coin, checks the DSDB and releases a set amount of money to the merchant in a single step.

Online vs. Offline E-cash Schemes

One important distinction that can be made about e-cash schemes is whether they are online or offline. An online e-cash scheme requires the bank (deposit counter) to be online and communicating with the merchant during the `Spend` protocol. There is therefore no advantage in not collapsing the `Spend` and `Deposit` protocols into a single `Spend-Deposit` protocol[1]. In particular, an online system immediately allows the merchant to check whether a coin has been doublespent.

An offline scheme does not require the merchant to communicate with the bank during the `Spend` protocol. The merchant can deposit coins later using the distinct `Deposit` protocol. If a coin is doublespent then this will be detected only when both coins are deposited. There must therefore exist a tracing mechanism to identify and punish users that doublespend coins, as the act of doublespending will not be identified during a transaction.

The Chaum architecture that we examine in this report is an online e-cash architecture[2]. It does not require user or merchant registration – indeed, one of its advantages is that it can be implemented without the merchant and user having known identities.

2 Security Requirements for an E-Cash Scheme

There are several high-level security requirements for an e-cash scheme. Formal security models for these requirements in the offline setting have been developed by Camenisch, Hohenberger and Lysyanskaya [1] but even these requirements have been criticised and may not be stable. There do not appear to be formal security requirements for online e-cash systems.

2.1 Correctness

If there is no malicious interaction in the protocol, then the protocol must run correctly. In other words, with very high probability, if the user correctly interacts with the bank to produce a coin and correctly interacts with the merchant to spend the coin, then the merchant should be able to interact with the bank to deposit the coin. A requirement of this security conditions is, for example, that there are enough serial numbers so that the probability that a user unintentionally creates the two coins with the same serial number is small.

[1] The opencoin project has investigated a situation in which it is advantageous for a user to spend a coin with a merchant without the presence of an online deposit counter and/or doublespend database. This mechanism will be briefly discussed in Section 5.

[2] Although, see footnote 1.

2.2 Anonymity

It should be possible for the bank and the merchant, even if they are working together, to link a coin being used in a `Spend` protocol to a coin that was received during a `Withdraw` protocol. This appears to be a hard notion to formalise.

2.3 Balance

No combination of user and merchant can successfully execute a `Deposit` protocol on a coin that the user does not own. This can be defined in two ways: either that a user must somehow commit to a serial number during the `Withdraw` protocol and cannot spend a coin with a serial number to which it has not committed, or the user and merchant cannot deposit a greater number of coins than have been produced withdrawn by that user.

It should be noted that this doesn't provide any protection for the merchant. In other words, it might be possible for a user to spend more coins than have been withdrawn. The merchant would not realise that the user was fraudulent until it attempted to deposit the coins.

In an online e-cash system, however, spending and depositing coins are synonymous operations and so this attack can never occur. However, it should be noted that in offline systems or in the case when an online system is adapted to allow a user to spend a coin without the presence of the deposit counter, then this type of attack should be considered.

2.4 Doublespend

No combination of users and merchants should be able to deposit a coin with the same serial number twice (without either detecting the doublespend during the `Spend-Deposit` protocol in an online system or identifying the fraudulent user or merchant in offline system).

2.5 Culpability

No combination of merchant and bank can frame an honest user for a doublespend. Note that if we have perfect detection of doublespent coins by an online e-cash system, then this requirement is unnecessary.

2.6 Fairness

No merchant (or other outside agency) can execute a `Withdraw` protocol with the bank on an account that does not belong to them. Furthermore, no merchant (or other outside agency) can cause a serial number to be placed on the doublespend database unless except as part of a legitimate `Deposit` protocol.

3 Chaum's Original Architecture

The architecture proposed by Chaum in 1982 [2] is as follows. Note that this architecture does not require that the user or the merchant register their identities with the bank. It makes use of a blind signature scheme.

- `Setup`: The central authority generates a master public/private key pair for a (traditional) signature scheme. This key pair is going to be used to certify the details of the scheme and the public keys of the mint. The public key is widely circulated.

  The central authority generates a series of public/private key pairs for the blind signature scheme for each denomination of coin that the mint can issue. These are widely distributed in certificates signed using the central authority's master private key.

- `Withdraw`: The user prepares a blank coin. This is a bitstring that contains information, in some commonly agreed format, on the identity of the bank, the denomination of the coin and a randomly chosen serial number.

  The user undertakes the blind signature protocol on the blank with the bank (mint). The user must also supply details of what denomination the coin is supposed to have and must somehow make payment for the coin. The mint performs a blind signature using the private key corresponding to the denomination indicated by the user.

  The complete coin consists of the blank and the signature on the blank.

- `Deposit-Spend`: The user sends the complete coin (blank and signature) to the merchant. The merchant checks that the signature verifies using the public key specified by the denomination of the coin in the blank. If not, the merchant rejects the coin.

  The merchant sends the coin to the bank (deposit counter). The bank checks that the signature verifies using the public key specified by the denomination of the coin in the blank. If not, the bank rejects the coin and informs the merchant of this fact.

  Otherwise the bank checks to see if the serial number of the coin exists on the doublespend database. If so, the bank rejects the coin and informs the merchant of this fact. If the serial number isn't on the doublespend database, then the bank adds the serial number to the database and accepts the coin. The merchant is informed of this acceptance and somehow receives payment for the coin.

This architecture suffers from an attack against the fairness of the scheme (see Section 2.6) in which the merchant prevents the user from spending a coin by placing the coin's serial number on the doublespend database. In this attack, the merchant receives the coin from the user and extracts the serial number. Then the merchant withdraws a coin from his own account with the same serial number and deposits the coin back into his account. This causes the serial number to be placed on the doublespend database. The merchant then attempts to complete the `Deposit-Spend` protocol with the user's original coin and (correctly) informs the user that the coin has already been spent. The user cannot recover the value of the coin.

4 Chaum's Encrypted Architecture

The opencoin team proposed a solution to this problem which involved encrypting part of the transaction between the user and the merchant using public-key encryption techniques. This solution was independently known to the literature. Its origin is uncertain but it certainly appears in the survey article on e-cash produced by Schoenmakers in 1997 [7].

The updated protocol is as follows:

- `Setup`: The central authority generates a master public/private key pair for a (traditional) signature scheme. This key pair is going to be used to certify the details of the scheme and the bank's other public keys. The public key is widely circulated.

  The bank generates a public/private key pair for a public-key encryption scheme to be used by the mint. The public key is widely distributed in a certificate signed using the central authority's master private key.

  The central authority generates a series of public/private key pairs for the blind signature scheme for each denomination of coin that the mint can issue. These are widely distributed in certificates signed using the central authority's master private key.

- `Withdraw`: The user prepares a blank coin. This is a bitstring that contains information, in some commonly agreed format, on the identity of the bank, the denomination of the coin and a randomly chosen serial number.

The user undertakes the blind signature protocol on the blank with the bank (mint). The user must also supply details of what denomination the coin is supposed to have and must somehow make payment for the coin. The mint performs a blind signature using the private key corresponding to the denomination indicated by the user.

The complete coin consists of the blank and the signature on the blank.

- `Deposit-Spend`: The user encrypts the complete coin (blank and signature) using the bank's public encryption key. This ciphertext is sent to the merchant. Note that the merchant cannot now check the signature on the blank.

  The merchant sends the encrypted coin to the bank (deposit counter). The bank decrypts this ciphertext and recovers the complete coin. The bank now checks that the signature verifies using the public key specified by the denomination of the coin in the blank. If not, the bank rejects the coin and informs the merchant of this fact.

  Otherwise the bank checks to see if the serial number of the coin exists on the doublespend database. If so, the bank rejects the coin and informs the merchant of this fact. If the serial number isn't on the doublespend database, then the bank adds the serial number to the database and accepts the coin. The merchant is informed of this acceptance and somehow receives payment for the coin.

This protocol prevents the previous attack where the merchant registers the serial number of the coin that the user has sent to him by encrypting the serial number using the bank's public encryption key.

We may now consider the security requirements for an e-cash scheme:

Correctness

If the blind signature scheme and the public key encryption scheme are correct (i.e. operate correctly) then it appears as if the e-cash scheme is correct.

Anonymity

If the blind signature scheme is secure, then it appears as if the e-cash scheme should provide anonymity for the user.

Balance

If the blind signature scheme is secure, then it appears as if the e-cash scheme should provide balance protection for the bank.

Doublespend

If the blind signature scheme is secure, it appears as if the e-cash scheme should prevent double-spending.

Culpability

Culpability is not a security requirement for online e-cash architectures.

Fairness

The merchant can always perform an attack against fairness, even in this improved architecture. Consider the following attack scenario. A user wishes to purchase an item that costs one coin and sends a valid coin to the merchant. The merchant submits this coin to the bank *twice* and reports to the user that the coin has been doublespent (a fact that the bank will be able to confirm). The user will not know that the merchant actually submitted the coin twice because the bank does not provide any indication to the user that a coin that has been successfully spent.

There are further problems that occur if the user pays for multiple items simultaneously. Since a coin is not bound to one particular purchase, the merchant may legitimately use a coin for

one purchase and then (illegally) use the coin for a second purchase. This results in the second purchase being declined as the coin has been doublespent and the merchant gaining knowledge of a large number of valid coins.

Schoenmakers [7] recommends that (the hash of some) transaction information about the purchase be included as part of the encrypted information that the bank receives; however, it is unclear if this solves the problem as an encryption scheme does not protect the integrity of transmitted information. Nor are the effects on anonymity clear for a situation in which the bank is supplied with transaction information. We conclude that more study is required to gain assurances about the fairness of the encrypted Chaum architecture.

We stress that these are only preliminary security assessments. No attempt has been made to formally prove the security of the architecture for any of these properties. A proof would not guarantee security but would be strong evidence that no subtle attacks have been missed in this short and preliminary analysis. Further study is required before we can conclude that the architecture is secure.

5 Partially Offline E-cash Schemes

The concept of a partially offline e-cash scheme has been discussed by the opencoin group. This situation would only occur when the merchant trusts the user to behave honestly. In such a situation, the merchant could accept the encrypted coin as part of a spend protocol and trust that it was correct. The merchant would only discover whether the coin was valid upon depositing the coin.

We note that the merchant would have to have complete faith in the user. In the original Chaum scheme, the only danger was that the user could provide a doublespent coin to the merchant, and the merchant would not discover the doublespend until depositing the coin.

In the encrypted Chaum scheme, the user can provide any encrypted text and the merchant will not be able to discover whether it is a coin or not. Furthermore, even if the merchant is somehow able to check whether the ciphertext contains a correctly signed coin or not, the user can repeatedly encrypt a single coin to create different ciphertexts all of which are the encryption of correctly signed coins.

This mode of operation of the scheme could only be used in situations where the merchant is prepared to accept full liability for the transaction.

6 Implementation Guidance

In this section, we give some advice for implementing the encrypted Chaum architecture:

6.1 Key Management

It is important that any cryptographic key follows a strict key lifecycle. Keys must be generated in such a way that their validity period (the times between which they can be used) is well known and enforced. Furthermore, any public key must only be used for a single purpose.

Therefore, we conclude that the bank must use independently generated keys for the master key pair, the encryption key pair, and for *each* denomination of minting key. It is easy to show, in the case of Chaum's blind signature scheme [2], that the use of one key for two purposes can break the entire security of the e-cash scheme.

It is notable that the doublespend database for a particular denomination can be purged after the public key used to verify coins of that denomination is completely removed from service.

It should also be noted that the bank should have a robust system for handling key revocation. There are several reasons why one of the bank's public keys may need to be revoked. For example, a key could be compromised or a doublespend database for a particular denomination could become too large and need to be purged.

There are several systems for public key revocation that could be used in the e-cash architecture. We note that, if one of the bank's public keys is to be revoked, then the users and merchants need to be informed of the fact in a timely manner. Most notably, since the users and merchants may not frequently undertake `Withdraw` or `Deposit` protocols, the bank cannot necessarily wait until they have a legitimate interaction with the user/merchant. This would suggest that some kind of offline certificate revocation scheme may be appropriate; however, such a scheme assumes the existence of a channel from the bank to the user/merchant which may not exist except during `Withdraw` and `Deposit` protocols.

Further research is required to establish the advantages and disadvantages of the various certificate revocation methods in an e-cash system.

6.2 Randomness

The generation of cryptographically secure random bits and random numbers is of critical importance to any security system. It is well-known that many of the standard random bit generation algorithms are not sufficiently secure to be used in cryptographic applications. The use of a standardised models for random bit generation is recommended [3].

It is possible to identify some of the problems that may occur if poor random bit generation is used. This list should not be considered exhaustive – other attacks may be possible besides those listed here:

- If poor random bit generation is used to generate the central authority's master key pair, or if the master private key becomes compromised, then the security of the entire scheme is compromised. The attacker may create new minting keys and encryption keys – the results of which are may be equivalent to the case where these keys are compromised.
- If poor random bit generation is used to generate the bank's encryption key pair, or if the encryption key pair becomes compromised, then a merchant may attack the fairness of the protocol using the attack described in Section 3. Furthermore, since the basic Chaum scheme (see Section 3) has not been analysed in this report, other attacks may be possible as well.
- If poor random bit generation is used to generate the bank's minting key pairs, or if a minting key pair becomes compromised, then an attacker can issue coins to himself without using the `Withdraw` protocol and exchange them for money with the bank. This is an attack against the balance property of the scheme.
- If the bank uses poor random bit generation techniques during the `Withdraw` protocol, then the security of the minting key used during the protocol is at risk. If the minting key is compromised, then the balance property of the scheme becomes compromised.
- If the user use poor random bit generation techniques during the `Withdraw` protocol, then the bank may be able to compromise the user's anonymity when the coin is deposited.
- If the user uses poor random bit generation techniques during the `Spend-Deposit` protocol (i.e. whilst it is encrypting the coin) then the merchant may be able to compromise the fairness of the scheme. Furthermore, since the basic Chaum scheme (See Section 3) has not been analysed in this report, other attacks may be possible as well.

There are no other points within the encrypted Chaum architecture in which randomness is used. However, randomness may also be used as part of the series of underlying protocols that secure the channels between the user, merchant and bank. The use of poor random bit generation for in these underlying protocols can cause a loss of confidentiality, integrity protection and authentication for these channels (see Section 6.4).

6.3 Encryption Algorithms

The scheme relies on the security of a public-key encryption algorithm. If a sufficiently secure scheme is not selected, then the merchant may be able to perform an attack against the fairness of the scheme. The use of a standardised secure encryption scheme is recommended [5]. Furthermore, due to the large amount of data that may need to be encrypted, a hybrid encryption scheme may be appropriate.

6.4 Channel Security

The analysis presented in this report assumes the use of confidential and integrity-protected channels whenever two parties are communicating. If this assumption is not correct, then the system may become vulnerable to attacks.

6.4.1 Confidentiality

There are several problems that may occur during the protocol if the confidentiality of the channel is broken. These include, but are not limited to, the following:

- If the confidentiality of the channel between the user and the bank is compromised during the `Withdraw` protocol, then an attacker may obtain the coin being supplied to the user. If this coin is spent by the attacker before it is spent by the user, then the user will never be able to spend the coin. This is an attack against fairness.
- If the confidentiality of the channel between the user and the merchant or the merchant and the bank is compromised during the `Spend-Deposit` protocol, then an attacker may obtain the encrypted coin. If the attacker can submit this coin to the bank before the merchant can submit it, then the user will never be able to spend the coin. This is an attack against fairness.

We note that confidentiality can be provided in a number of ways, including the use of encryption algorithms and by routing controls (i.e. only routing data through secure devices). This latter system may be useful when a user obtains their coins from a trusted unit in a bank or shop (e.g. by inserting a smartcard designed to contain the coins into the device).

If encryption is used, then it is recommended that encryption schemes that are secure against active attacks are used [4, 5, 6].

6.4.2 Integrity Protection

There are several problems that may occur during the protocol if the integrity of the channel is broken. These include, but are not limited to, the following:

- If the integrity of the channel between the user and the bank is compromised during the `Withdraw` protocol, then the attacker may be able to alter the final message from the bank to the user. This would result in the user receiving an unusable coin, but the bank still accepting payment for the coin. This is an attack against the fairness of the scheme.
- If the integrity of the channel between the merchant and the bank is compromised during the `Spend-Deposit` protocol, then the attacker may alter the message from bank to the merchant which tells the merchant whether the protocol has been successful or not. This could cause a doublespent coin to be declared as valid and causes the merchant to be defrauded.

6.4.3 Authentication Protection

There are several problems that may occur if the entities on the channels do not authenticate their identities to each other. Most notable is the possibility for man-in-the-middle attacks, which may

compromise the integrity protection and confidentiality of the channel. Other attacks, including denial of service, may be possible.

It is clear to see that it is possible for the bank to authenticate to the user and the merchant. However, it is difficult to see how the user and/or merchant can authenticate themselves to the bank without having some digital identity. Further research should be done to determine the extent to which the problem compromises the security of the e-cash scheme and whether any solutions to this problem can be found.

6.5 Error Messages

The security of cryptographic protocols can only be considered in some (formal or informal) cryptographic model. A protocol whose security is considered sound in some model may not be secure if it is used in an environment which does not conform to the model. This is particularly true if the environment gives more information to an attacker than is available to the attacker in the model.

A classic implementation problem of this kind is associated with the use of error messages. If a cryptographic protocol should fail for some cryptographic purpose (for example, the failure of a hash function check or a value being out of a specified range) then the cryptographic protocol should not reveal the reason for the protocol failure[3].

This consideration may be of particular importance when one considers the security of the confidential channels between the entities in the protocol. In such a situation, depending upon the particular algorithms being used, the attacker might be assumed not to be able to determine any information about the protocol messages being passed. If the protocol is able to observe any information about a message corresponding to a particular ciphertext (for example, whether it is correctly formatted or whether it gives rise to an error message) then it may be possible to break the confidentiality of the scheme.

The exact security requirements for the encrypted Chaum architecture are unclear and further research is needed to determine the extent to which error messages can be used within the system.

References

[1] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In R. Cramer, editor, *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer-Verlag, 2005.

[2] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology – Crypto '82*, pages 199–204. Plenum Publishing, 1982.

[3] International Organisation for Standardisation. *ISO/IEC 18031, Information technology — Security techniques — Random bit generation*, 2005.

[4] International Organization for Standardization. *ISO/IEC 18033–3, Information technology — Security techniques — Encryption Algorithms — Part 3: Block Ciphers*, 2005.

[5] International Organization for Standardization. *ISO/IEC 18033–2, Information technology — Security techniques — Encryption Algorithms — Part 2: Asymmetric Ciphers*, 2006.

[6] International Organization for Standardization. *ISO/IEC FCD 19772, Information technology — Security techniques — Authenticated encryption*, 2007.

[3] It is possible for the protocol to reveal the source of an error if the attacker could determine that the protocol would fail from the input data themselves. For example, if a protocol involved passing unencrypted data that included a digital signature of a publicly known value, then it would be reasonable to send an error message stating that the digital signature was not valid, because the attacker could determine if the digital signature was valid without interacting with the protocol.

[7] B. Schoenmakers. Basic security of the ecash$^{\text{TM}}$ payment system. In B. Preneel and V. Rijmen, editors, *State of the Art in Applied Cryptography*, volume 1528 of *Lecture Notes in Computer Science*, pages 338–352. Springer-Verlag, 1997.