

Counting Abuses Using Flexible Off-line Credentials

No Author Given

No Institute Given

Abstract. Mobile and ad-hoc networks allow businesses to provide a new range of applications and services and at the same time they introduce new constraints that have important effects on the way in which security primitives must be designed. This is challenging because it translates to a demand of richer and more flexible security primitives that often need to satisfy stricter requirements than traditional wired networks scenarios. In this paper we focus on one of this primitive, namely security credentials. We present a solution that extends the existing protocols used to implement off-line credentials such that, not only abuses can be detected but they can also be counted. Our solution addresses the problem of 1-time and 2-times credentials and we will conclude by discussing the challenges that need to be solved to generalize the primitive to $k = n$.

1 Introduction

With the advent of mobile and ad-hoc networks (MANETs), security researchers face new challenges due to the change of some of the assumptions underlining the security solutions designed for conventional wired networks such as Internet. Global connectivity for example, is replaced by local connectivity, so in MANETs it is not reasonable to assume always the connectivity to a central server or to a trusted third party.

On the other hand, MANETs provide an innovative technology that could potentially enable a wide spectrum of new applications and services. The ability to carry around credentials that can be used at *any time* and at *any place* will increase the opportunities for merchants to sell more services and for customers to buy services following convenient and easy procedures.

Of course, any solution to be acceptable must prevent customers from abusing the use of such credentials (i.e. spending them more times than legitimate, forging them, etc.) and at the same time must prevent merchants from forging credentials and to make them appear as if they were genuinely spent by customers. Besides and not less important, the privacy of customers need to be protected (i.e., merchants cannot record spending habits of customers).

Starting with Chaum [3], researchers proposed several solutions to implement credentials that can be used off-line. Some of those solutions try to detect while others to prevent possible abuses. Some employ single-use credentials, other allow the same credential to be used many times up to a configurable limit. All the

existing schemes however, cannot detect how many times the abuse occurred. This would be a very useful feature to have since existing solutions have to implement fixed penalty schemes that cannot distinguish how many times the customer has abused her credential.

In this paper, we propose a solution called *Flexible Off-Line Credentials* that allows off-line use of credentials and detects not only that the credential has been abused (that means it has been used more than the legitimate k threshold) but also how many times the abuse has occurred. We propose and analyse solutions for the case $k = 1$ and $k = 2$ and then we discuss why the extension to $k = n$ would require further work. Our solution allows fair fine-based schemes that punish abusers proportionally to their abuse. Our scheme can also be combined with a credit-based scheme to provide a single flexible mechanism that offers the advantages of both debit and credit-based solutions without the usability limitations typically of the debit-based approach.

The rest of this paper is organized as follows. Next section summarizes the related work. Section 3 gives the preliminaries for our work. Section 4 explains our protocol for flexible one-time credentials. Section 5 discusses how to extend this protocol for credentials that can be used k -times. Section 6 and Section 7 are for security and performance analysis of the proposed protocols respectively. Section 8 concludes the paper.

2 Related Work

The problem of implementing private credentials that cannot be abused in the sense we described above, can be easily solved if the system can rely upon a central authority that both issues credentials and later monitors and stores in a database all these credentials as they are used [1]. So when a customer presents her credential to a merchant, the merchant can check with the authority if that credential is still valid and can be accepted, preventing any abuse. Such approach cannot be used with MANETs because the access to the database cannot be guaranteed.

Alternatively Chaum [2], proposes an off-line method that uses special hardware trusted by the issuer of the credentials that prevents customers to abuse the use of credentials. Such approach however, cannot be used in many contexts where deploying such hardware is simply too expensive.

Credit-based solutions can be also used off-line. Think for instance credit cards. Customers only need to present an authentic nonrepudiable piece of information (signed credit card voucher¹). to be able to buy a service from a merchant. The problem with this approach is that abuses of credentials may not be promptly detected if an online server is not available (it is a general practice for merchants to check the validity of the credit card in real-time with an online server instead of relying on an offline list of revoked credit card numbers). Besides, this solution does not provide anonymity at all since the credit card

¹ If the cardholder is not required to sign such a voucher, she has simply no protection against fraud.

information given to the merchant is bounded with the identity of the customer to allow redemption.

Debit-based solutions use credentials that can be used off-line, too (similar to paper cash). The customer has to decide beforehand how many times she is going to use the credentials and she pays in advance for their use at the time they are issued rather than when (and if) they are used. After payment she is given credentials which can be used anonymously like spending paper cash in physical world. However unlike paper cash, in digital world credentials are simply a sequence of bits and can easily be copied and spent at different merchants illegitimately.

Two approaches given above (online server and trusted hardware) aim at preventing these abuses. However, prevention is not always necessary, and we will argue also not desirable, provided that the solutions could guarantee detection of credentials' abuses. The third and last approach is not to prevent the customer overspend her credentials but detect and punish her afterwards. The user's anonymity is guaranteed if she follows the rules of the game but once she tries to cheat, her identity is learned and her abuse is proved to redeem the losses. Chaum [3], was the first to introduce a solution of this type applied to digital cash as credentials. Of course the aim there was to prevent double spending. Brands [4] proposes a more efficient solution and also extends Chaum's work to the case of credentials that can be used k -times and detects if this threshold is abused by the costumer. Another notable example of k -times credentials is due to Camenisch and Lysyanskaya [9]. But none of these approaches has the functionality of detecting how many times the user has overspent her credential. Up to our best knowledge, this paper is the first to propose a flexible credential protocol in this respect.

In a recent work, Bussard and Molva [5] have revisited the idea of offline credentials for MANETs scenarios. They propose a solution where the costumer is deterred from abusing the credential by the threat of a fixed fine if an abuse is detected. However the solution cannot distinguish if the credential has been abused once, ten times or one hundred times leading to possible unfair penalties. Beside their solution does not provide privacy for the costumers.

3 Preliminaries

In this section we provide some background information on the primitives used in our protocol.

3.1 One-Time Signatures

We will summarize the operation of one-time signature (OTS) construction proposed by Lamport [6] as follows:

As in traditional signing, the message is hashed to get a fixed reduced size of b bits if needed ($b = 160$ if SHA-1 is used). Then the signer generates $2b$ random numbers of size sufficiently long (so called pre-images). He then computes the

hash of each random number and predistributes them (usually called as hash-images) securely to the intended verifier(s) as the one-time public key. Now there are two random numbers associated for each bit of the message to be signed. If the value of the bit is 0, the signer reveals the first of these random numbers, if it is 1, he reveals the other one. The whole bunch of revealed random numbers constitutes the OTS. The receivers can easily verify the OTS by calculating the hash of each pre-image revealed and checking whether the result is equal to the corresponding hash-image in the one-time public key.

Reducing the size of the one-time public key is possible. At the extreme, the signer can compute the hash of all hash-images and sends this single hash value as the one-time public key. Now the signing procedure needs to be modified, as an OTS the signer reveals not only the pre-image values as in the previous case but also the hash values of the pre-images which are not revealed as a part of signature. Unless this is done, obviously the receiver can not verify the OTS.

Lamport's scheme is severely unoptimized hence there are a lot of previous work for optimization (e.g. [7]). But we do not explain them here since they are not relevant to our purposes in this paper.

3.2 Blind Signatures

Blind signatures, first introduced by Chaum [8], allow someone (say Alice) to get a message signed by some other party (say Bob) without revealing any information about the message to him. It is straightforward to implement blind signatures using RSA since it has the multiplication commutative property. Here is how blind RSA works (all operations are in modulo n):

1. Alice generates a random number r , computes the hash of his message $h(m)$ and supplies $h(m) * r^d$ to Bob (d is Bob's public key). Multiplying with r^d serves for the purpose of "blinding" the message.
2. Bob signs what Alice has supplied with his private key e and returns $[h(m)]^e * r^{(ed=1)}$ back to Alice.
3. Alice divides this value with r to get Bob's signature on $h(m)$.

Even when Bob receives back his own signature, he can not figure out when he has signed it. This provides unconditional privacy for Alice since her identity can not be traced when she gives Bob's signature to someone else.

3.3 Cut-and-Choose Method

Above, we explained how blind signatures can be used to sign something without seeing it. On the other hand in some applications before signing, Bob needs to ensure that the content of the message should have some appropriate format. This requirement seems to be contradicting with the requirement of not seeing the message however fortunately there is a technique called cut-and-choose to achieve this functionality.

The basic idea is as follows. Suppose Alice wants Bob to sign n messages. She does not prepare n blinded messages but $n + m$ of them. Upon receipt of these blinded messages, Bob chooses a random subset having m elements and requests Alice to reveal the r values for these m messages. If these unblinded messages have the appropriate format, it is highly probable that Alice has also prepared appropriately the blinded remaining of n messages. Then Bob signs them and sends back to Alice. In other words, Bob *cuts* m messages and *chooses* the remaining n messages to sign. Alice can then perform the unblinding and use the signatures as desired.

Note that there is only a probabilistic guarantee that the signed messages have the appropriate format. Alice can cheat by conforming to the format for the m messages but not for one of the remaining n . However the cheating probability can be made negligible by choosing appropriate m and n values.

4 The Proposed Protocol

In this section we introduce our protocol when the credentials are valid for one-time. We first introduce the presumed environment where the proposed protocol is to be employed. Then, we describe initialization and operation of the protocol and illustrate how the exact number of multiple spending can be detected. Finally, we show how this feature can be used for seamless conversion from debit to credit based solutions.

4.1 System Model and Assumptions

We envision a pervasive environment where servers either fixed or mobile provide different kinds of services to users. Users are mobile, they can visit these servers to request for the service. To ease the management, the servers are grouped into zones. In each zone there is one authorization authority (AA) to which the servers have registered. Users first contact to AA or the credential issuer (CI) which has made an agreement with AA in order to get their "one-time credentials" which is then used to request service from one of registered servers.

To make the model more concrete, consider the following example. Let us suppose that Plane Airline signed an agreement with all the shops that are in the Airport of Hangar such that every Plane's customer is entitled of a discount of 5 dollars she can use at any shop of the airport upon presentation of her Plane's boarding card. Shops within the airport may not be connected to each other and they are not always connected with a central server (i.e. the company that manage the airport). This is a quite reasonable assumption since connectivity has a cost thus normally shops prefer to deal with off-line transactions. The airport is managed by a company that sells services to the airport's users. It sublets premises to shopkeepers and to the airlines, it provides basic services such as power, water, security, etc.

In our example, customer buys, following the existing procedures, the flight ticket from Plane. At the time of check-in, Plane issues to the customer a boarding card carrying the one-time credential. The solution must detect customers

who use the credential more than once (e.g., by simply presenting the credential to different shops). The solution must also prevent to forge credential otherwise they can claim false compensation.

Note that the credentials were redeemed by the shops at the company that manage the airport, and not at the airline server. This is because the company managing the airport is the one who has made the agreement with Plane. This company then checks whether there is any illegitimate spending of the same credential (twice or more). In case of abuse it can ask the airline for the abuser details collected when she bought the ticket.

The Hangar Airport constitutes the zone and each shop inside it are the servers. The company that manages the airport and Plane Airline are referred as AA and CI, respectively. All Plane's customers are potential users. We assume there is no continuous on-line connectivity between AA and servers but they can exchange some data periodically (e.g. weekly or daily basis). AA and CI can also have a periodic communication possibly less frequently (e.g. monthly basis).

After getting their "one-time credentials", the users do not have any connection to CI. Their only interaction is with the servers to request for the service.

In this model as a requirement the privacy of users is guaranteed if they have used their credential legitimately only once. The key issue is assuring that users use their one-time credentials only once but not more. Our solution is based on the "postponed punishment" principle as in [5] and most offline e-cash schemes. While he obtains his credential, every user also signs a contract stating that his identity will be learned and some money will be charged from him if CI can later prove that the user has used his one-time credential more than once.

Unlike previous work using our protocol, CI (and AA) can determine the exact number of multiple uses therefore while preparing the contract it becomes possible to set the amount of penalty as a function of number of multiple uses (e.g. 5 euros for double spending, 10 euros for triple spending and so on).

The final assumption we make in designing the protocol is that initially AA sets the maximum number of servers that it can support and distributes this piece of information to CI. This assumption seems reasonable because it does not imply that servers can not dynamically registers to or leaves from AA. The only constraints is not to have more than the fixed maximum number of servers registered at the same time.

4.2 Initialization

To start with, AA assigns identification numbers (ID-s) to registered servers in a way unique to our protocol. The IDs are expressed as binary vectors having zeros in all coordinates except one and the single '1' should appear in a different coordinate for all the vectors. Let us give an example to illustrate the concept:

Example: Suppose the maximum number of servers is set up to be 5 and number of servers registered is currently 3. One way of assigning IDs is as follows:

00001 to 1st server
00010 to 2nd server
00100 to 3rd server

As you see each ID has only one nonzero coordinate. It is easy to assign other IDs conforming to this rule if the maximum limit is not exceeded.

While the servers are registered, AA also sends securely CI's public key in order to let the servers verify the signatures. After IDs are assigned, to get a one-time credential from CI, the user executes the following steps:

1. The user learns the maximum number of registered servers. Suppose the maximum is b .
2. The user prepares $4b$ random numbers. He then divides these random numbers into two groups having $2b$ elements each.
Example: If the maximum limit is 5, the user prepares 20 random numbers.
 Group 1: $x_1, x_2, x_3, \dots, x_{10}$
 Group 2: $y_1, y_2, y_3, \dots, y_{10}$
3. The user calculates the hash of each of these random numbers (to generate hash images).
4. From both groups, the user retrieves the hashes having the same sequence number ($h(x_i)$ and $h(y_i)$) and concatenates them. He then generates random numbers (r_i 's) and prepares $2b$ blinded messages as follows:

$$B_i = (h(x_i) \parallel h(y_i)) * (r_i)^d \text{ mod } n$$

5. The user also calculates the hash of concatenation of two random numbers having the same sequence number i.e. $C_i = h(x_i \parallel y_i)$. We refer C_i 's as *conca - hashes*.
6. The user sends B_i 's and C_i 's to CI. Blinding is not needed for conca-hashes.
7. For a successful operation, the user should prepare the hash images and do the encoding of all conca-hashes (C_i 's) honestly as described above. To verify this, CI uses the cut-and-choose method. More precisely, for a random subset having b elements out of $2b$, CI asks the user to prove that the hash images are correctly prepared and conca-hashes are correctly encoded.
8. As a proof, the user reveals random numbers (r_i 's) and pre-images (x_i and y_i 's) for the subset CI asked for.
9. CI verifies the subset of blinded messages (B_i 's) and conca-hashes (C_i 's). It then signs the remaining subset of blinded messages as follows (to simplify notation, let's assume the remaining "chosen" subset has sequence numbers from 1 to b):

$$S = \prod_{1 \leq i \leq b} (B_i)^e \text{ mod } n$$

- The user first performs the unblinding (by dividing S with $\prod_{1 \leq i \leq b} r_i$) and then reindexes lexicographically the rest of "chosen" hash-images (of which the user did not reveal the corresponding pre-images) as follows:

$$(h(x_1) \parallel h(y_1)) < (h(x_2) \parallel h(y_2)) < \dots < (h(x_b) \parallel h(y_b))$$

$$P = \prod_{1 \leq i \leq b} (h(x_i) \parallel h(y_i))$$

P constitutes the one-time public key certified by CI's signature. The signature on the one-time public key is in fact the one-time credential issued by CI for the user.

Example: When the number of server is limited to a maximum of 5, the one-time credential has the following format:

One-time Credential = $Signature_{CI}[(h(x_1) \parallel h(y_1)) * (h(x_2) \parallel h(y_2)) * (h(x_3) \parallel h(y_3)) * (h(x_4) \parallel h(y_4)) * (h(x_5) \parallel h(y_5))]$

- In order to establish the penalty for multiple use, the user and CI prepares a contract. In the contract, the "chosen" conca-hashes are listed after the re-indexing. Again we illustrate this idea with an example:

Example Penalty Contract: This contract is prepared between Plane Airline and user Alice. The conca-hashes are as follows: C_1, C_2, C_3, C_4, C_5 . By signing this contract, user Alice accepts to pay X euros for the first two conca-hashes Plane Airline will be able to show its encoding (reveals x_i and y_i such that $C_i = h(x_i \parallel y_i)$) and X euros for the encoding of each extra conca-hash after the first two.

At a first glance, it looks strange to require two conca-hashes for the first pay and one extra for the succeeding ones. We defer justifying this to the end of this section.

4.3 Operation

In the initialization, the user gets his one-time credential from CI. We now want to show how he can use this credential to get a service from a registered server:

- The user learns the ID of the server he would like to get a service from.
- The user sends the server his one-time credential and one-time public key. He then signs the ID of the server using the method described in subsection 2.2.

Example: Suppose the server's ID is 00001. The user signs this ID by revealing the following chunk:

$x_1, x_2, x_3, x_4, h(x_5)$ and

$h(y_1), h(y_2), h(y_3), h(y_4), y_5$

(He reveals x_i and $h(y_i)$ if the i -th bit value is 0 and y_i and $h(x_i)$ if it is 1).

3. The server verifies one-time credential using CI's public key, checks the lexicographic order of hash images (i.e. $(h(x_1) \parallel h(y_1)) < (h(x_2) \parallel h(y_2)) < \dots < (h(x_5) \parallel h(y_5))$) and verifies one-time signature using one-time public key of the user. The server also stores the credential and the signature.

4.4 Detecting the Number of Multiple Uses

In case when the user wants to issue the same one-time credential to the same server, this can easily be detected and avoided by the server. So this is not a real issue here. The more critical cheating is the one where the user uses his one-time credential to get a service from a different server. This can not be detected by the server instantly but later AA can detect it after collecting the one-time credentials from the servers.

Example: Besides server 1, suppose the user reuses his one-time credential also to server 2 with the ID of 00010. He should sign the server's ID by revealing

$$x_1, x_2, x_3, h(x_4), x_5 \text{ and} \\ h(y_1), h(y_2), h(y_3), y_4, h(y_5)$$

At the end of the day, AA can easily detect that the one-time credential has been used twice because the same pre-images has appeared in two different signatures (e.g. x_1, x_2, x_3). If AA is also the credential issuer, it can also prove this by showing the encoding of conca-hashes. If CI is another entity, AA should ask CI for the redemption by sending the signatures.

After receiving two one-time signatures from AA, CI can show the encoding of C_4 and C_5 since both x_4 and y_4 as well as x_5 and y_5 are obtained as parts of signatures. Due to the signed contract, showing the encoding of C_4 and C_5 would let CI to get the first penalty pay of X euros.

Note that the identity of the cheater is written on the signed contract and learned by CI. However depending on the security policy it might be preferred to protect his privacy from AA and other third parties.

This is not the whole story. AA can detect and CI can prove not only double spending but also subsequent spendings by the same logic.

Example: After spending it to server 1 and 2, suppose now before the day ends the user reuses his one-time credential to server 3 with the ID of 00100. After collecting the one-time signatures, CI can also show the encoding of C_3 and therefore the user should pay an extra amount of X euros.

4.5 Seamless Conversion from Debit to Credit-based Scheme

Since our protocol can count the number of times credentials are being used, it provides the possibility to convert it from a debit to a credit scheme without additional requirements. Mechanisms that cannot quantify abuses clearly have to detect such abuses with the aim of preventing them. In our case, this is not necessarily true and the capability of using the credentials even more than a pre-established threshold can be useful provided that there is an exact counting. A practical example where this can be useful is the following. Let us consider

the case of metro tickets valid for one trip. The holder of the ticket, because he is in a hurry, does not have coins, or the ticket office is closed cannot buy a new ticket. However at the moment in which he validates the old ticket to have access to the train, the validating machine will detect the abuse and it let him to pass through by simply charging him with a premium. This can also occur more than once, may be with a premium that increases with the number of times. Many customers will be happy to pay a bit more than the normal price, but much less than the fine, for this flexibility. Thus, the same protocol can be converted from a debit to credit based one. Of course the underlying assumption here, as in any credit-base scheme, is that the system has access to the customer details to credit for the premium.

5 Extension for 2-times Credentials

In the previous section, we proposed a flexible protocol for credentials to be used for one-time. The next step is to extend our protocol for k -times credentials. Such credentials could be used for many applications; implementing digital tram tickets valid for multiple trips (i.e. as the Dutch "strippenkaart") is an example. A straightforward method to extend this protocol for k -times credentials is to issue k different one-time credentials that would be spent separately one by one. Unfortunately this straightforward extension has efficiency problems especially with respect to storage on the user's device and computation required in the initialization phase. Hence it would be wise to explore if a more efficient extension is possible. This section is devoted for this purpose.

For one-time credentials, the server IDs are binary vectors and there are two random numbers committed for each coordinate of the ID space. The protocol works as desired because the IDs are assigned in a way that forces the user to reveal both of the random numbers for a fixed number of coordinates in case of multiple spending (two coordinates for the second spending and one more for subsequent spendings). Based on these observations, below we list the requirements for a more general case of k -times credentials:

1. The server IDs should be $k + 1$ -ary vectors and there should be $k + 1$ random numbers committed for each coordinate.
2. When the credential is used for $k + 1$ times, all of the random numbers should be revealed for a deterministic number of coordinates (preferably only one coordinate for convenience).
3. When the credential is used for more than $k + 1$ times, the number of coordinates for which all the random numbers are revealed should be a one-to-one function of how many times the credential has been used.

As the final requirement, all these should be satisfied with the minimum length of server IDs because of efficiency reasons. We find an efficient solution only for the specific case of $k = 2$, as we will explain in the rest of this section.

5.1 An ID Assignment Algorithm for 2-times Credentials

To satisfy the requirements listed above, the most critical issue is the assignment of server IDs. Pseudocode for the algorithm to assign IDs for 2-times credentials is as follows:

```
Input(max) /* read the maximum limit for the number of servers */
for  $i = 0$  to max-1 do begin
    for  $j = 0$  to max-3 do begin /* IDs have length of max-2 each */
        server[i][j]:='e' /* 'e' means empty */
    end
end
for  $i_1 = 0$  to max-3 do begin
    for  $i_2 = i_1+1$  to max-2; do begin
        for  $i_3 = i_2+1$  to max-1 do begin
            for  $j = 0$  to max-3 do begin
                if server[ $i_1$ ][ $j$ ]== 'e' then server[ $i_1$ ][ $j$ ]:='0'
                if server[ $i_2$ ][ $j$ ]== 'e' then server[ $i_2$ ][ $j$ ]:='1'
                if server[ $i_3$ ][ $j$ ]== 'e' then server[ $i_3$ ][ $j$ ]:='2'
                if server[ $i_1$ ][ $j$ ]  $\neq$  server[ $i_2$ ][ $j$ ]  $\neq$  server[ $i_3$ ][ $j$ ] then break
            end
        end
    end
end
end
```

The rationale of this algorithm is to consider every subset of combinations with three elements and assign a different ternary value of each element in every subset for only one coordinate (while keeping the number of coordinates required to a minimum). As an example, let us consider the case of maximum number of servers set to be 7 as shown in Table 1. Note that each of the 5 coordinates is assigned a ternary value (a value in the range of 0 to 2). You can experiment with this table by trying different combinations of spendings to confirm that this ID assignment satisfies the requirements listed above i.e. when the credential is spent for the third time all random numbers are revealed for only one coordinate. Additionally, for subsequent spendings all the random numbers are revealed for exactly one more coordinate.

5.2 Initialization of the Protocol for 2-times Credentials

After server IDs are assigned, similar to earlier case the following steps should be executed:

1. The user learns the length of server IDs (which is equal to the maximum number of registered servers minus two). Suppose the length is b .
2. The user prepares $6b$ random numbers and divided them into three groups. The user also calculates the hash of these random numbers.

coordinate	#5	#4	#3	#2	#1
server #1	0	0	0	0	0
server #2	0	0	0	0	1
server #3	0	0	0	1	2
server #4	0	0	1	2	2
server #5	0	1	2	2	2
server #6	1	2	2	2	2
server #7	2	2	2	2	2

Table 1. Assignment of Server IDs for 2-times credentials when max=7.

3. Instead of two, the user concatenates three hashes and prepapes $2b$ blinded messages as follows.

$$B_i = (h(x_i) \parallel h(y_i) \parallel h(z_i)) * (r_i)^d \bmod n$$

4. The conca-hashes the user prepares also has three random numbers as the input: $C_i = h(x_i \parallel y_i \parallel z_i)$
5. The rest of the initialization strictly follows the earlier case. Instead of two, three random numbers are committed for each coordinate.

5.3 Using 2-times Credentials

In one-time credentials, the coordinates of server IDs have binary values therefore per each coordinate depending on its value the user should reveal one of two random numbers and the other's hash value. Now since the coordinates have ternary values the user should reveal one of three random numbers and the other two hash values. Let's see an example to have a better understanding:

Example: Suppose the server's ID is 00012 (server #3). The user signs this ID by revealing the following chunk:

$x_1, x_2, x_3, h(x_4), h(x_5)$ and
 $h(y_1), h(y_2), h(y_3), y_4, h(y_5)$ and
 $h(z_1), h(z_2), h(z_3), h(z_4), z_5$

(He reveals $x_i, h(y_i)$ and $h(z_i)$ if the i-th bit value is 0; $y_i, h(x_i)$ and $h(z_i)$ if it is 1; and $z_i, h(x_i)$ and $h(y_i)$ if it is 2).

5.4 Remarks

Our solution for 2-times credentials has however, some limitations and if used for $k > 2$ it became less efficient than using k different one-time credentials.

- When the credential is valid for more than one-time, the user might want to use it (legitimately) for the same server however if there is only one ID

assigned to each server, there is no way to achieve this securely with the extension proposed. Fortunately there is a simple trick that enables us to do that. The trick is to assign two IDs instead of one to each server. On the other hand, this trick obviously makes the protocol less efficient due to increase in the ID space required.

- Modifying the ID assignment algorithm for three-times credentials is possible. However as the number of server increases three-times credentials rapidly become less efficient than three one-time credentials. This is due to the non-linear increase in the size of ID space required. The formula for calculating the number of coordinates as a function of number of servers is given below for three-times credentials:

$$\#Coordinates = \sum_{i=4}^{max} (i - 3)$$

Due to the inefficiency of our ID assignment algorithm for more than 2-times credentials, at the moment it is recommended to use a combination of 2-times credentials and one-time credentials to issue credentials valid for more than two times. For instance for a three-times credentials, one 2-times credentials and one one-time credentials can be used. To support our claim, we will make the performance comparison of different choices for three-times credentials in section 7.

- One nice feature of the protocols proposed in this paper is the preservation of user privacy. Even the credential issuer can not link the legitimate spendings to the identity of users. There is a stronger requirement than privacy called **unlinkability** that means inability to link one spending with another one without considering two spendings remain anonymous or not. Since the extension for 2-times credential described in this section does not provide unlinkability, one-time credentials should be used as a building block to implement k -times credentials to satisfy the unlinkability requirement.

6 Security Analysis

In the following, we consider the basic security requirements imposed on an off-line credential. We also show that our protocol satisfies the requirements, therefore it is secure.

6.1 Unforgeability

For a secure operation, the credential should not be forged. To forge it, an adversary can attempt to:

- forge CI’s public key signature
- find pre-image(s) such that the hash of it is equal to the one of the hash images CI has certified.

Clearly the first attack is not specific to our protocol. The second attack is infeasible if the hash function is pre-image resistant.

6.2 Proving Multiple Use

In subsection 4.4, we have shown how overspending can be proven by CI. However there is a condition for this proof. If the encoding of conca-hashes was not performed as described, the proof fails. Just like earlier work (e.g., [3, 5]) our protocol offers only probabilistic guarantees because of using cut-and-choose method. Let us show the calculation of probability of an undetected cheating as a function of protocol parameter b (length of server IDs).

In step 7 of the initialization phase in subsection 4.2, CI chooses b out of $2b$ conca-hashes. There are $C(2b, b)$ subsets in total that CI can choose from (C denotes combination where $C(2b, b) = (2b)!/b!b!$).

Cheating by the user would not be detected only if the user truly did the encoding of only b out of $2b$ conca-hashes and CI has chosen exactly this single subset. The probability of choosing this subset is

$$p(c) = \frac{1}{C(2b, b)}$$

For instance this probability is approximately equal to 0.004 if $b = 5$. As b increases, the probability of undetected cheating decreases.

6.3 Revocation of CI's Public Key

One of the biggest hassles in public key cryptography is the issue of public key revocation. Most of the protocols depending on public key cryptography should deal with it in some way or other and our protocol is not an exception. Our protocol relies on the assumption that when for some reason CI's public key needs to be revoked, this piece of information is immediately passed through the registered servers in **push-based** manner rather than using a pull-based approach such as CRL or OCSP. Similarly it is also possible to revoke users' credentials. To minimize the need to revoke the credentials, it is useful to set a short expiration time of all credentials issued.

7 Performance Analysis

In this section, we would like to compare the performances of the proposed protocol and its extensions. For this purpose, as an example we consider the case where a three-times credential is issued for the user. There are three implementation alternatives for this case; either 3 one-time credentials, or 1 one-time credential and 1 2-times credential, or 1 three-times credential is issued.

Since the user's device might be a constrained device as opposed to CI's or AA's, we choose to compare the storage and computational requirements on the user's device. The most heavy computation is the blinding operation since it involves public key operation that's why for the sake of simplicity we omit other computations in Table 2. Note that the second most resource consuming operation is likely to be the random number generation which is directly proportional to the storage requirement given in Table 2.

	Storage	Blinded Messages
3 one-time	$12L * max + 3sig$	$6 * max$
1 two-times + 1 one-time	$10L * max - 12 + 2sig$	$4 * max - 4$
1 three-times	$8L * \sum_{i=4}^{max} (i - 3) + sig$	$2 * \sum_{i=4}^{max} (i - 3)$

Table 2. Performance comparison of proposed protocol and its extensions

In Table 2, L is the length of random numbers and hash images (we assume they have the same length), max is the maximum number of registered servers, and sig denotes the length of public key signature. As seen from this table, using a combination of one-time and 2-times credentials is more efficient than using only one-time credentials. One other observation is that when maximum number of servers exceeds a threshold, issuing 1 one-time credential and 1 two-times credential performs better than the three-times credential. We calculate the threshold value as 7 for storage and 8 for blinded message requirements. This result demonstrates the need for future work for efficient solutions for $k > 2$.

Note that one other parameter that might be a bottleneck for constrained environments is the bandwidth required in the real-time interaction between the user and AA. When this bandwidth requirement is considered, we see that using one-time credentials performs better because of the fact that credentials are independent from each other and can be spent separately.

As a final note, we observe that our protocol is more round-efficient than earlier protocols discussed in Section 2. This is because by fixing the user’s response to each server using constant IDs, a two-round challenge-response protocol is not needed. We do not go further in comparing the performances of protocols since they have different features as we have already mentioned.

8 Conclusion and Future Work

We described a new approach to privacy-preserving limited-time off-line credentials. Our approach uses the postponed punishment principle just like earlier work but it also has the unique feature of detecting exactly how many times the user has overspent his credential. This gives us the ability to have a single framework incorporating both debit-based and credit-based solutions as well as the flexibility to dynamically adjust the penalty for overspending.

Our protocol is clearly not appropriate for global Internet, it is rather designed for pervasive environments where number of servers is limited.

As a future work, it is promising to explore possibilities to have a more efficient extension of our protocol to the general case of k -times credentials.

References

1. D. Chaum: Online cash checks. In Advances in Cryptology. Proceedings of EURO-CRYPT’89 (LNCS 434), pages 288-293. Springer-Verlag, 1990.

2. D. Chaum, T.P. Pedersen: Wallet Databases with Observers. In Advances in Cryptology. Proceedings of CRYPTO'92 (LNCS 740), pages 89-105. Springer-Verlag, 1993.
3. D. Chaum, A. Fiat, M. Naor: Untraceable electronic cash. In Advances in Cryptology. Proceedings of CRYPTO'88 (LNCS 403), pages 319-327. Springer-Verlag, 1990.
4. S. Brands: A technical overview of digital credentials. Research Report, February 2002.
5. Laurent Bussard, Refik Molva: One-Time Capabilities for Authorizations without Trust. Proceedings of Second IEEE International Conference on Pervasive Computing and Communications(PerCom 2004), Orlando, Florida, March 14-17, 2004.
6. L. Lamport: Constructing digital signatures from a one-way function. Technical report SRI-CS-98. SRI International Computer Science Laboratory, October 1979.
7. Kemal Bicakci, Gene Tsudik, Brian Tung: How to construct optimal one-time signatures. Computer Networks (Elsevier), Vol.43(3), pp. 339-349, October 2003.
8. D. Chaum: Blind signatures for untraceable payments. Proceedings of CRYPTO'83, pages 199-203.
9. Jan Camenisch and Anna Lysyanskaya: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. Proceedings of EUROCRYPT 2001, LNCS 2045. pages 93-118, Springer-Verlag, 2001.