Anonymous NIZK Proofs of Knowledge with Preprocessing

Stefano D'Amiano* and Giovanni Di Crescenzo**

Abstract. In this extended abstract we present an unpublished result in [6] which extends a result in [4]. We give a non-interactive zero-knowledge proof system of knowledge with preprocessing, whose main property is that, after executing two preprocessing phases and given a transcript of a proof phase, the verifier is not able to relate the transcript to any of the two preprocessing phases significantly better than random guessing. The technique used has motivated the cash scheme in [3]. Because of this result, only mentioned but used in [3], the main observation of Pfitzmann et al. in [8] against the cash scheme in [3] doesn't hold. We also discuss the other observations of Pfitzmann et al. in [8] against the cash schemes in [3, 5] and show that *all* of them don't hold. As a conclusion, the cash schemes in [3, 5] are not broken at all.

1 The result

A non-interactive zero-knowledge (nizk) proof system of knowledge with preprocessing is a pair of protocols (preprocessing,proof) between two parties P and V in which, after an interactive preprocessing, P can send with a single message to V a zero-knowledge proof that P knows a witness to the truth of a certain statement. Such a system can be used in an electronic cash system by a spender to prove to a shop the knowledge of a signature of a coin released by the bank. Informally, we will say that a nizk proof system with preprocessing is *anonymous* if any verifier V', after running a pair of preprocessing protocols, and given the transcript of any proof protocol, cannot relate the proof protocol to one of the two preprocessing protocols with probability significantly greater than $1/2$.

Our result says that the nizk proof of knowledge with preprocessing given in [4] for all NP languages can be made anonymous. Let us briefly recall the system (A,B) in [4]. Let σ be a public random reference string. In the preprocessing phase the prover A computes a commitment *com* to a string s using coins r and interactively proves the statement H:'I know s and r such that *com* is a commitment to s, using coins r'. The verifier B verifies this proof. Now, let L be any NP-complete language, and let (x, w) be a pair (instance,witness) such that the prover wants to prove that he knows the witness w such that $x \in L$. In order to prove this, A first computes $\beta = w \oplus f_s(x)$, where f_s is a pseudo-random function, and $\oplus$ is the bitwise logical xor operator. Then A gives a nizk proof $proof_T$ (of membership) on the public random reference string σ of the

* Computer Science Department, Cornell University, Ithaca, NY, USA
** Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, 92093

L.C. Guillou and J.-J. Quisquater (Eds.): Advances in Cryptology - EUROCRYPT '95, LNCS 921, pp. 413-416, 1995.

statement T: 'there exist strings r, s, w such that *com* is a commitment to s using coins r, w is a witness for $x \in L$, and $\beta = w \oplus f_s(x)$'. In order to verify this proof, B needs the string *com* obtained in the preprocessing phase. The view of B in the preprocessing is $(com, H, proof_H)$ and a transcript of the proof phase will be $(\beta, \sigma, T, proof_T)$. Clearly, here the statement T used in the proof phase contains the commitment *com* used in the preprocessing phase. Thus, after running two commitment phases, given the two views $(com_1, H_1, proof_{H_1})$ and $(com_2, H_2, proof_{H_2})$ and the transcript $(\beta, \sigma, T, proof_T)$ of a proof phase, B can check if the commitment used in T is com_1 or com_2 and thus associate the proof with exactly one of the two preprocessing run.

Our extension is simply stated; our proof system (P,V) is the same as the above (A,B), with the following two modifications. First, in the preprocessing phase V also signs the commitment *com* and sends the signature sig_{com} to P. Second, in the proof phase, instead of the above statement T, the prover uses the statement T': 'there exist strings r, s, w, com, sig_{com} such that *com* is a commitment to s using coins r, w is a witness for $x \in L$, $\beta = w \oplus f_s(x)$ and sig_{com} is a correct signature of *com*'. Then the view of V is: $(com, sig_{com}, H, proof_H)$ in the preprocessing phase, and $(\beta, \sigma, T', proof_{T'})$ in the proof phase. Now the two views are not related, thus a transcript of a proof phase cannot be matched with a view in the preprocessing phase. The formal proof is based on the following idea: assume by contradiction that there is an efficient non-uniform verifier V', which, after running two preprocessing phases $prep_1$ and $prep_2$, and getting a transcript pf of a proof phase, can compute $i \in \{1, 2\}$ such that pf is the proof associated to the preprocessing $prep_i$. Then, it is possible to use V' in order to construct an efficient algorithm A which efficiently opens commitments or contradicts the zero-knowledgeness of the proof system used, or contradicts the pseudo-random property of the function used.

Finally, we remark that this technique has been used in the cash scheme in [3] in the construction of the withdrawing and the spending protocols.

2 The observations by Pfitzmann et al.

In the paper [8], Pfitzmann et al. make some observations against the cash scheme in [2] [3]. Now, we consider all the observations in [8] and show that *all* of them do not hold.

The main untraceability flaw: In their first observation, Pfitzmann et al. state, without proof, that our protocol in [2] doesn't satisfy the untraceability requirement because by receiving a proof of knowledge in a deposit protocol, the bank can link it to the user who run the preprocessing when opening his account. Also, they write '*This is a general problem with NIZKP with preprocessing: At least as long as preprocessing is a 2-party protocol, any reference to it, i.e., any proof, identifies the two parties...*'. In [3] the protocol presented in previous section is

[3] This version was the result of a few-days two-paper merging; unclearly, the proceedings version [3] has never been asked.

used, with only a sketch of description, as the same technique extending [4] is used in the construction of the protocols for withdrawing a coin and for spending a coin in the electronic cash scheme. As the above proof system is anonymous, the observation by Pfitzmann et al. is wrong. In particular we stress that it is not true that nizk proofs with preprocessing cannot be used in cash schemes. [4]

Weakness of the definition of untraceability: Here Pfitzmann et al. criticize our definition of untraceability, as it does not require untraceability against payees. We answer by noticing that in our paper, after giving the protocols for withdrawing a coin and spending a coin, we discuss in Section 6 that, by simply applying one of the many secret-sharing based techniques for avoiding the double-spending of a coin, we easily obtain a cash system. This scheme clearly satisfies also untraceability against payees (but it doesn't give transferability!). However, we don't prove such an easy statement, and thus we don't need a definition for it. Thus we investigate transferability of coins and we obtain the first cash scheme in which coins can be transferred without increase in size. This result was not known. Untraceability against payees is not required in it: on the other hand, if it was, then such result would not be possible, as proved in [1].

An untraceability flaw in double-spending detection: Here Pfitzmann et al. write *'Until the double-spender is detected, the bank cannot prove that a coin was double spent, i.e., the users have to believe this. Thus, a dishonest bank could always claim that the coin was deposited twice in order to get the payees to disclose the payers'.* This is not true, as in our protocol, the bank discovers that a double-spending has occurred from the fact that she receives two signatures of a same coin; in fact, in Section 6 we write *'the bank broadcasts a message stating that a double-spending of c has occurred ... to prove this, she writes on the public file the two different signatures of c'.*

Finding the wrong double-spender: A first observation by Pfitzmann et al. here is: a user *'may have received the coin twice, e.g., from an attacker at different times. In this case, the honest user is likely to be found before the attacker and punished as the double-spender.'* This is not true; in fact, when we discuss the single spending requirement, we never say that the first user found to have spent twice the coin is punished. We say that *'the bank uses the signatures received to reconstruct the two paths ... that have been taken ... by the coin c'*; this clearly means that the bank will first receive all signatures and then she will reconstruct the whole paths taken by the coin, where the double-spender is the source of these paths. A second observation here is a situation in which a honest user H_1 spends twice the same coin as he received it twice. They say *'H_1 has two completely identical signatures from A_1, so she cannot prove she got the coin twice and is punished'.* Here Pfitzmann et al. assume that for signature schemes constructed under general cryptographic tools, two signatures on the same messages are equal. This is clearly not true (see [7] and references therein),

[4] Also, when presenting the system in [2], they observe that *'The protocol in [AmCr] is ambiguous about whether the coin must be passed'.* In [2] actually it is written *'send C, $Proof$ to U_2'* instead of *'send c, $Proof$ to U_2'*; however no other occurrences are in the paper for C (except for Chaum).

and actually if it was true, then the results in [7] would prove that nizk proofs of membership are equivalent to general cryptographic tools, i.e., one-way functions (an open problem about nizk proofs).

No divisibility together with transferability: Here it is observed that complexity parameters are not reduced in our divisibility scheme. This is not true, in fact in Section 5 we write *'...to spend a part of c, say, of value 2^{k-h}, a user U_j gives only the random string used for the commitment at a node at level h'*. We stress that the most expensive complexity parameter used in our construction is just the number of random bits used in commitments, (it is usually one order bigger than the others).

Achieving weak untraceability more easily: Here an alternative solution to obtain untraceability with respect to the bank is suggested by Pfitzmann et al. Unfortunately, the suggested solution doesn't work: a coalition of the user who is depositing the coin and the bank is enough to make the coin traceable.

The system presented at CIAC 94: We also discuss the three observations about the cash scheme in [5]. The first is that the bank cannot announce if it rejects the withdrawal. We have always thought that there are many easy techniques for doing this, all depending from implementation; for instance, the bank can erase the coin published by the withdrawer. The second observation is that there is no untraceability. Of course our protocol does not satisfy a more general definition of untraceability than that given in the paper! This definition is weaker than usual just in order to obtain the non-interactivity of the protocol (which otherwise seems very hard to get). However, we observe that such a definition is not very weak; for instance, real-life cash satisfies it. The third observation seems the only correct so far: the deposit scheme can be attacked by a chosen-message attack. However, it is not clear which utility this attack could give, and also the scheme is easy to repair: the user that deposits the coin just proves to know a signature, instead of giving it (there exist such proofs in literature).

References

1. D. Chaum and T. Pedersen, *Transferred Cash Grows in Size*, Eurocrypt 92.
2. S. D'Amiano and G. Di Crescenzo, *Methodology for Digital Money based on General Cryptographic Tools*, Preproceedings of Eurocrypt 94.
3. S. D'Amiano and G. Di Crescenzo, *Methodology for Digital Money based on General Cryptographic Tools*, to appear on Proceedings of Eurocrypt 94.
4. A. De Santis and G. Persiano, *Communication Efficient Zero-Knowledge Proof of knowledge (with Application to Electronic Cash)*, STACS 92.
5. G. Di Crescenzo, *A Non-Interactive Electronic Cash System*, CIAC 94.
6. G. Di Crescenzo, *Anonymous NIZK Proofs of Knowledge with Preprocessing*, manuscript, December 1993.
7. S. Goldwasser and R. Ostrovsky, *Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent*, Crypto 92.
8. B. Pfitzmann, M. Schunter, and M. Waidner, *How to break another provably secure payment system*, Eurocrypt 95.