On the Impossibility of Batch Update for Cryptographic Accumulators

Philippe Camacho
Dept. of Computer Science, University of Chile,
Blanco Encalada 2120, 3er piso, Santiago, Chile.
pcamacho@dcc.uchile.cl

December 15, 2009

Abstract

A cryptographic accumulator is a scheme where a set of elements is represented by a single short value. This value, along with another value called witness allows to prove membership into the set. In their survey on accumulators [FN02], Fazzio and Nicolisi noted that the Camenisch and Lysyanskaya's construction[CL02] was such that the time to update a witness after m changes to the accumulated value was proportional to m. They posed the question whether *batch update* was possible, namely if it was possible to build a cryptographic accumulator where the time to update witnesses is independent from the number of changes in the accumulated set.

Recently, Wang et al. answered positively by giving a construction for an accumulator with batch update in [WWP07, WWP08]. In this work we show that the construction is not secure by exhibiting an attack. Moreover, we prove it cannot be fixed. If the accumulated value has been updated m times, then the time to update a witness must be at least $\Omega(m)$ in the worst case.

1 Introduction

An accumulator is a scheme that hashes elements of a set X into a single, constant size value called the *accumulated value*. Then it is possible to prove membership into X of an element x using a verification algorithm that takes this accumulated value, an element x, and some value called *witness*. The first construction of such scheme is due to Benaloh and De Mare [BdM94]. Several improvements later followed, especially in the work of Camenisch and Lysanskaya [CL02] who showed how to build dynamic accumulators (where the accumulated set can evolve), and how to use them as a tool to efficiently implement anonymous credentials. In their survey on accumulators [FN02], Fazio and Nicolisi pointed out that, in the Camenisch and Lysyanskaya's construction, the time to recompute the witnesses once the accumulated set has been modified was proportional to m, the number of changes of the accumulated value. This raised a natural question: *"Is it possible to construct dynamic accumulators in which the update of several witnesses can be performed using a constant size update information?"*

Wang et al. [WWP07] answered positively this question by showing a construction that allows batch update. Unfortunately, we show that this construction and its improvement [WWP08] are not secure. Moreover, we prove that there is no way to fix Wang's et al. accumulator by giving a lower bound of $\Omega(m)$ in the time that takes to update a witnesses after m updates (delete operations) to the accumulated set.

Related Work. As mentioned above, the first construction for cryptographic accumulators with batch updates was given in [WWP07] and later revised in [WWP08]. Even though no other constructions are known, the existence of accumulators with batch update seems to have been taken for granted, and in fact referenced in subsequent work. Damgärd and Triandopoulos [DT08] cite their availability as an example of a accumulator construction based on the Paillier cryptosystem. Camenisch et al. [CKS09] also mentionned Wang et al.'s construction.

Organization of the paper. First, in Section 2, we briefly recall the notion of a dynamic accumulator, which we follow with our attack. The more general impossibility result is then presented in section 4. We conclude in section 5.

2 Dynamic Accumulators

In this section we recall general definitions about dynamic accumulators.

Syntax: Accumulator schemes consider two types of participants: a *manager* who initializes the parameters, computes the accumulated value and the witnesses, and a *user*, whose role is to verify the membership of elements in the set and possibly ask for elements insertion/deletion.

Definition 1. *(Adapted from [FN02]) Let k be the security parameter. An accumulator scheme $\mathfrak{Acc}$ consists of the following algorithms:*

- $\mathsf{Setup}(1^k)$*: a probabilistic algorithm that takes the security parameter k in unary as argument and returns a pair of public and private key (PK, SK) as well as the initial accumulated value for the empty set $Acc_\emptyset$.*
- $\mathsf{Eval}(X, Acc_\emptyset, PK, [SK])$*: given a finite set of elements X, a public key (or the private key), and the initial accumulated value $Acc_\emptyset$, this algorithm returns the accumulated value Acc_X corresponding to the set X.*
- $\mathsf{Verify}(x, w, Acc, PK)$*: given an element x, a witness w, an accumulated value Acc and a public key PK, this deterministic algorithm returns OK if the verification is successful, meaning that $x \in X$, or $\perp$ otherwise.*
- $\mathsf{Witness}(x, Acc, PK, [SK])$*: this algorithm returns a witness w associated to the element x of the set represented by Acc.*
- $\mathsf{Insert}(x, Acc_X, PK, [SK])$*: this algorithm computes the new accumulated $Acc_{X\cup\{x\}}$ value obtained after the insertion of x into set X.*
- $\mathsf{Delete}(x, Acc_X, PK, [SK])$*: this algorithm computes the new accumulated value $Acc_{X\setminus\{x\}}$ obtained from removing the element x from the accumulated set X.*
- $\mathsf{UpdWit}(x, w, Acc_X, Upd_{X,X'}, PK)$*: this algorithm recomputes the witnesses for some element x that remains in the set. It takes as parameters the element x whose new witness must be updated, the "old" witness w with respect to previous set X represented by the accumulated value Acc_X, some update information $Upd_{X,X'}$, the current set X', and the public key PK, and returns a new witness for x. We remark that this algorithm is run by the user.*

The above definition is slightly more general than the one by Camenisch and Lysyanskaya [CL02] as it does not depend on how these algorithms are implemented, and it explicitly includes the update algorithm UpdWit in the syntax of a scheme. We remark that our impossibility

result do not depend on whether the algorithms Verify, Witness, Eval, Insert, Delete, UpdWit are deterministic or probabilistic.

CORRECTNESS: The correctness property of an accumulator scheme is rather straightforward: if an element x belongs to the accumulated set X and if the corresponding witness w has been computed using Witness then the verification process should pass.

Definition 2. *(Correctness) Let X be a set, Acc_X its associated accumulated value and $x \in X$. An accumulator scheme is correct if* $\mathsf{Verify}(x, w, Acc_X, PK) = OK$ *where Acc_X is the accumulated value of the set X, $x \in X$, PK is the public key and w is the witness obtained by running* Witness *on Acc_X, x and PK.*

SECURITY: Consider an experiment where the adversary plays the role of a user and attempts to forge a witness (i.e. finding a valid witness for an element that does not belong to the set) while having access to an oracle that implements the operations done by the manager.

Definition 3. *([CL02]) Let $\mathfrak{Acc}$ be an accumulator scheme. We consider the notion of security denoted $\mathcal{UF}$-$\mathcal{ACC}$ described by the following experiment: given an integer k, the security parameter, the adversary has access an oracle $\mathcal{O}$ that replies to queries by playing the role of the accumulator manager. Using the oracle, the adversary can insert and delete a polynomial number of elements of his choice. The oracle replies with the new accumulated value. The adversary can also ask for witness computations or updates. Finally, the adversary is required to output a pair (x, w). The advantage of the adversary $\mathcal{A}$ is defined by:*

$$Adv^{\mathcal{UF}\text{-}\mathcal{ACC}}_{\mathfrak{Acc}}(\mathcal{A}) \quad = \quad \Pr\left[\, \mathsf{Verify}(x, w, Acc, PK) = 1 \wedge x \notin X \,\right]$$

where PK is the public key generated by Setup.

In the following section we show why Wang et al.'s construction is not secure.

3 An attack on the Accumulator with Batch Update of Wang et. al [WWP07, WWP08]

3.1 Problems with the proof

A security proof for the scheme was presented[1] in the original paper by Wang et al. [WWP07]. Unfortunately, there are two main problems in the security proof.

First the adversary $\mathcal{B}$ used there appears to run the **KeyGen** algorithm which means it knows the factorization of the modulus n, or at least the knowledge of $\Phi(n^2)$ and $\lambda = lcm(p-1, q-1)$ where n is a safe modulus of the form $n = pq$ with p, q primes since $\beta = \sigma\lambda mod(n^2)$. In fact, without β computed in such a way, the correctness of the construction cannot hold anymore. (The value σ is also required.) Therefore, it is not clear how the reduction to break the *es-RSA assumption* can be achieved.

The second problem is that, to break the es-RSA assumption, $\mathcal{B}$ needs to find non trivial (y, s) such that $y^s = x \ mod \ n^2$ where x is given as input to $\mathcal{B}$. This value x does not seem to be mentioned in the proof.

[1]The subsequent paper [WWP08] fixes a correctness flaw in [WWP07] but does not present a new security proof. The attack we consider, however, also works for the improved version [WWP08].

3.2 Description of the attack

As to show that the construction is not secure, i.e., the proof of security cannot be fixed, we present an attack. This attack considers the updated scheme presented in [WWP08].

The idea is simply to delete an element from the set, and then update the witness of this element with the update information obtained by the execution of the algorithm **DelEle**. We can then check easily that this new witness is a valid one for the deleted element, which of course should not happen.

So we start with the set $X = \{c_1\}$ for some c_1. We have $x_1 = F(c_1^\lambda \ mod \ n^2) \ mod \ n$. Then a random element c_* is chosen and $x_* = F(c_*^\lambda \ mod \ n^2) \ mod \ n$ is computed. The accumulated value is set to $v = \sigma(x_1 + x_*) \ mod \ n$. The witness value $W_1 = (w_1, t_1)$ for c_1 is defined by $w_1 = a_c c_1^{-t_1\beta^{-1}} \ mod \ n^2$ where $a_c = y_* y_1 \ mod \ n^2$, $y_1 = c_1^{\lambda\sigma\beta^{-1}} \ mod \ n^2$, $y_* = c_*^{\lambda\sigma\beta^{-1}} \ mod \ n^2$, and t_1 is random.

Then the adversary asks the manager to delete element c_1. This means that the new accumulated value is $v' = v - \sigma x_1 + \sigma(x_{**}) \ mod \ n = \sigma(x_* + x_{**}) \ mod \ n$ where $x_{**} = F(c_{**}^\lambda \ mod \ n^2) \ mod \ n$ and c_{**} is random. The auxiliary value a_u used to update the witnesses is $a_u = y_{**} y_1^{-1} \ mod \ n^2$ where $y_{**} = c_{**}^{\lambda\sigma\beta^{-1}} \ mod \ n^2$. So updating the witness w_1 with a_u we obtain $w_1' = a_u w_1 \ mod \ n^2 = y_{**} y_1^{-1} y_* y_1 c_1^{-t_1\beta^{-1}} \ mod \ n^2 = y_{**} y_* c_1^{-t_1\beta^{-1}} \ mod \ n^2$. Then $w_1'^\beta c_1^{t_1} \equiv (y_{**} y_* c_1^{-t_1\beta^{-1}})^\beta c_1^{t_1} \ mod \ n^2 = (y_{**} y_*)^\beta \ mod \ n^2 = (c_{**} c_*)^{\lambda\sigma}$. It follows that

$$\begin{aligned} F(w_1'^\beta c_1^{t_1} \ mod \ n^2) &\equiv F((c_{**} c_*)^{\lambda\sigma} \ mod \ n^2) \ mod \ n \\ &\equiv \sigma(F(c_{**}^\lambda \ mod \ n^2) + F(c_*^\lambda \ mod \ n^2)) \ mod \ n \\ &\equiv \sigma(x_* + x_{**}) \ mod \ n \\ &\equiv v' \ mod \ n \end{aligned}$$

This shows that (w_1', t_1) is a valid witness for the deleted element x_1. Therefore the scheme is not secure. Indeed the problem is simply that the information a_u allows to update *every* old witnesses including w_1 for which such an update should not be possible.

4 A lower bound for updating the witnesses

The attack of the last section is an indication that the proposed construction may have some design flaws. In this section, we show that the problem indeed is more fundamental and the batch update is essentially unrealizable. We argue this by presenting a lower bound on the size of $Upd_{X,X'}$, the information needed to update the witnesses after m changes (more precisely deletions). Any update algorithm UpdWit must at least read $Upd_{X,X'}$, and so it also bounds the running time of any such algorithm.

Theorem 1. *For an update involving m delete operations in a set of n elements, the size of the information $Upd_{X,X'}$ required by the algorithm* UpdWit *while keeping the dynamic accumulator secure is $\Omega(m \log \frac{n}{m})$. In particular if $m = \frac{n}{2}$ with n even, we have $|Upd_{X,X'}| = \Omega(m)$.*

Proof. We consider the following scenario. The set accumulated in some point of the time is $X = \{x_1, x_2, ..., x_n\}$, and the corresponding accumulated value is Acc_X. We suppose the user possesses all the witnesses for each element in X and knows the accumulated value. Then m Delete operations are performed, that is the new set obtained is $X' = X - X_d$ where $X_d = \{x_{i_1}, x_{i_2}, ..., x_{i_m}\}$. The manager computes the new accumulated value $Acc_{X'}$ and sends it to the user along with the update information $Upd_{X,X'}$ required to update all the witnesses.

With this information $Upd_{X,X'}$, the user is able to reconstruct the set X_d of deleted elements by checking for each element of x, if its corresponding witness can be successfully updated using

$Upd_{X,X'}$ and the algorithm $\mathsf{UpdWitness}$. That is if the result of $w' = \mathsf{UpdWitness}(x, w, Acc_X, Upd_{X,X'}, PK)$ is not a valid witness i.e. $\mathsf{Verify}(x, w', X) = \perp$, then the element has been deleted. Note that this condition is necessary as in the contrary the scheme would be incorrect (some existing elements in X' could not have their witness updated) or insecure (it could be possible to compute witnesses for deleted elements).

Hence, as the user is able to recompute the set of deleted elements X_d only using the value $Upd_{X,X'}$, $Upd_{X,X'}$ must contain at least the information required to code a subset with m elements of a set with n elements. There are $\binom{n}{m}$ such subsets, so the minimum quantity of information required is $\log \binom{n}{m}$ bits. A standard lower bound [CLRS01] for the binomial coefficient is $\binom{n}{m} \geq (\frac{n}{m})^m$. Then we have $\log(\binom{n}{m}) \geq m \log \frac{n}{m}$. □

Corollary 1. *Cryptographic accumulators with* batch update *do not exist.*

5 Conclusion

This result shows that the batch update property as proposed in [FN02] cannot be obtained, as the time to update all the witnesses cannot be linear in the security parameter k, i.e $O(k)$, but it must be at least $O(m) = O(n) = O(p(k)) = \omega(k)$ for some polynomial p. Notice that our lower bound is not tight since Camenisch and Lysyanskaya's accumulator requires $O(p(k) \cdot k)$ time to update the witnesses after $O(p(k))$ changes. Nonetheless, in principle, it leaves some (potential) room to improve their construction by at most a factor of k.

Finally, one may consider getting around this impossibility result by not allowing deletions in the set. Unfortunately, such an accumulator can be trivially implemented by signing the elements of the set, as in this case there is no replay-attack. The witness for every element consists in its signature under the manager's private key, and clearly need not to be updated.

References

[BdM94] Josh Benaloh and Michael de Mare. One-way accumulators: a decentralized alternative to digital signatures. In *Proceedings of EUROCRYPT*, volume 1440, pages 274–285. Springer-Verlag New York, Inc., 1994.

[CKS09] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500. Springer, 2009.

[CL02] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of CRYPTO*, volume 2442, pages 61–76, 2002.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, September 2001.

[DT08] I. Damgård and N. Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538, 2008.

[FN02] Nelly Fazio and Antonio Nicolisi. Cryptographic accumulators: Definitions, constructions and applications. Technical report, 2002.

[WWP07] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. A new dynamic accumulator for batch updates. In *Information and Communications Security*, volume 4861, pages 98–112, 2007.

[WWP08] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. Improvement of a dynamic accumulator at icics 07 and its application in multi-user keyword-based retrieval on encrypted data. *Asia-Pacific Conference on Services Computing. 2006 IEEE*, 0:1381–1386, 2008.