Performances of Cryptographic Accumulators

Amrit Kumar, Pascal Lafourcade, Cédric Lauradoux

Performances of Cryptographic Accumulators

Amrit Kumar[1], Pascal Lafourcade[2], and Cédric Lauradoux[1]

[1] INRIA Rhône-Alpes
Grenoble, France
firstname.lastname@inria.fr
[2] LIMOS
Université d'Auvergne
Clermont Ferrand, France
pascal.lafourcade@imag.fr

Abstract. Cryptographic accumulators are space/time efficient data structures used to verify if a value belongs to a set. They have found many applications in networking and distributed systems since their introduction by Benaloh and de Mare in 1993. Despite this popularity, there is currently no performance evaluation of the different existing designs. Symmetric and asymmetric accumulators are used likewise without any particular argument to support either of the design. We aim to establish the speed of each design and their application's domains in terms of their size and the size of the values.

Keywords: Cryptographic accumulators, Bloom filter, Performance

1 Introduction

Cryptographic accumulators are space/time efficient data structures that are used to test if a value/element belongs to a given set. They are the cryptographic counterpart of a data structure very popular in the field of networking: the Bloom filters [5, 7].

Similar to a one-way hash function, (*asymmetric*) cryptographic accumulators generate a fixed-size digest representing an arbitrarily large set of values. Interestingly, a one-way accumulator further provides a fixed-size witness for any value of the set, which can be used together with the accumulated digest to verify membership of a value in the set. The security requirement being that the verification should fail for every value not in the set.

The first cryptographic accumulator by Benaloh and de Mare [2] is constructed using one-way RSA function $x^y \mod N$ for a suitably chosen N. Secure Bloom filters can be considered to be a *symmetric* variant of cryptographic

This is an extended and full version of the paper to appear in 39$^{\text{th}}$ IEEE Conference on Local Computer Networks (LCN), 2014.

This work has been partially supported by the LabEx PERSYVAL-Lab : n$^{\text{o}}$ ANR-11-LABX-0025.

accumulators *i.e.* which do not require witness for membership testing. A more precise and formal description, together with other constructions based on elliptic curves [16] and bi-linear pairings [9, 21], are developed in the next section.

Since their first appearance as a cryptographic primitive, accumulators have received attention in both directions: designing efficient and dynamic primitives and providing novel applications in different domains. Many applications have been found ranging from time-stamping schemes [20], search on encrypted data [4,13] to data aggregation in sensor networks [19] and distillation codes [16] among others. For instance, Sander *et al.* [26] use RSA accumulator to design blind and auditable membership proof scheme in the context of e-cash. The idea is that an accumulator of all the valid electronic coins is created and published. During coins transfer, anyone can verify the validity of the exchange thanks to the accumulator.

In many works, the choice of the accumulator is not motivated: it is therefore hard to know if the proposed solution is the most appropriate one. For instance, Li [19] uses a variant of HMAC based Bloom filter to securely aggregate events from sensors. A sensor upon detecting an unexpected event informs an aggregator station by sending a Bloom filter. The aggregator then merges the filters received and forwards it to the base station. In an earlier work, Zachary [31] uses RSA accumulator to detect unauthorized nodes in a sensor network. Indeed any other accumulator can potentially replace RSA accumulator as the only goal in this scenario is to test set-membership.

A full review of the performance of both symmetric and asymmetric accumulators with varied levels of security is clearly missing and is the motivation of our work.

Contributions Our goal is to evaluate the performances of existing cryptographic accumulators. Our main contributions are:

- A survey of existing asymmetric and symmetric accumulators.
- Software performance evaluation to determine which accumulator offers the best verification time. We also analyze the impact of implicit parameters (in particular hash functions) which affect efficiency.

From our benchmarks, we observe that elliptic curve cryptography (ECC) based accumulators have the best verification time followed by secure Bloom filters and lastly RSA. The former two are affected by the length of the values accumulated while RSA is immune to this. More details are provided in the following sections of the paper.

Outline In Section 2, we recall the definitions of cryptographic accumulators and the different existing designs proposed in the literature. In Section 3, we describe the setting and the softwares used for our evaluations. In Section 4, we analyze each accumulator separately and compare them.

2 Background on Cryptographic Accumulators

We first give cryptographic recalls concerning accumulators, then we present existing works on asymmetric and symmetric accumulators.

2.1 Definitions

Informally, an accumulator is a space/time efficient algorithmic solution to the *set-membership problem*, which consists in determining if a value belongs to a set. This set of values is often represented by a compact data structure such that for each value of the set it is possible to compute a *witness* that determines if the value is incorporated in the accumulator.

The notion of cryptographic accumulator, or accumulator for short, was first coined by Benaloh and de Mare in the seminal work [2]. The accumulators in this work are defined as a family of *one-way hash functions* (Definition 1) which satisfy an additional *quasi-commutative* property (Definition 2).

Definition 1 (One-way hash functions [2]). *A* family of one-way hash functions *is an infinite set of functions* $\mathcal{H}_\lambda := \{h_u : X_u \times Y_u \to Z_u\}$ *having the following properties:*

1. *For security parameter* λ *and each integer* u, $h_u(x, y)$ *is computable in time polynomial in the parameter* λ *for all* $x \in X_u$ *and for all* $y \in Y_u$.
2. *For any probabilistic polynomial-time algorithm* $\mathcal{A}$ *the following probability is negligible in* λ*:*

$$Pr[h_u \xleftarrow{\$} \mathcal{H}_\lambda; y, y' \xleftarrow{\$} Y_u; x \xleftarrow{\$} X_u; x' \leftarrow \mathcal{A}(1^\lambda, x, y, y') : h_u(x, y) = h_u(x', y')]$$

We recall that $f : \mathbb{N} \to \mathbb{R}$ is a *negligible* function in the parameter n, if for every positive polynomial p, there exists N, such that for all $n > N$, we have $f(n) < \frac{1}{p(n)}$.

Definition 2 (Quasi-commutativity [2]). *A function* $f : X \times Y \to X$ *is* quasi-commutative *if:*

$$\forall x \in X, \forall y_1, y_2 \in Y \ f(f(x, y_1), y_2) = f(f(x, y_2), y_1)$$

A one-way quasi-commutative function provides a primitive to design an accumulator. A function $f : X \times Y \to X$ is chosen depending on some security parameter together with an initial value $x \in X$. A set of values $Y' = \{y_1, y_2, \ldots, y_n\} \subset Y$ accumulates to a value z using the following equations $1 \leq i \leq n$:

$$\begin{cases} z_0 = x \\ z_i = f(z_{i-1}, y_i) \\ z = z_n \end{cases}$$

A witness w_i for a value y_i is the accumulated value for the set $Y' \setminus \{y_i\}$. Clearly, given a value y_i, and its corresponding witness w_i, one can easily verify if y_i had been accumulated by verifying the equality $z = f(w_i, y_i)$, which holds due to the quasi-commutativity of f.

2.2 State of the Art

We present the asymmetric accumulators before explaining the symmetric ones.

2.2.1 Asymmetric Accumulators The first accumulator proposed by Benaloh and de Mare [2] is asymmetric *i.e.* requires witness for verification. The construction uses modular exponentiation $f(x, y) = x^y \mod N$ as a one-way quasi-commutative function since it satisfies:

$$f(f(x, y_1), y_2) = (x^{y_1})^{y_2} = (x^{y_2})^{y_1} = f(f(x, y_2), y_1)$$

For exponentiation to be used for one-way accumulators, the modulus is chosen to be a product of two *safe* primes p and q of equal size. A prime p is safe if $\frac{p-1}{2}$ is also a prime number. A malicious attacker knowing the accumulated value z may try to forge a witness w for a randomly chosen value y by finding an initial value x verifying $x^y \mod N = z$. However, this is infeasible under the *RSA assumption*, Definition 3.

Definition 3 (RSA assumption). *When the modulus N is sufficiently large and randomly generated, and the exponent y and a value z are given, it is hard to compute x satisfying $x^y \mod N = z$.*

However, as informally noticed in [2] and later recognized by Nyberg in [22], one-wayness imposed in the definition might not suffice for certain applications where an adversary has access to the list of values to be accumulated. To remedy, one should consider a stronger property called *strong one-wayness* (Definition 4) where the attacker is not imposed the choice of y' as in Definition 1.

Definition 4 (Strong one-way hash functions [2]). *A* family of strong one-way hash functions *is an infinite set of functions $\mathcal{H}_\lambda := \{h_u : X_u \times Y_u \to Z_u\}$ having the following properties:*

1. *For security parameter λ and each integer u, $h_u(x, y)$ is computable in time polynomial in the parameter λ for all $x \in X_u$ and for all $y \in Y_u$.*
2. *For any probabilistic polynomial-time algorithm $\mathcal{A}$ the following probability is negligible:*

$$Pr[h_u \xleftarrow{\$} \mathcal{H}_\lambda; y \xleftarrow{\$} Y_u; x \xleftarrow{\$} X_u; x', y' \leftarrow \mathcal{A}(1^\lambda, x, y) : h_u(x, y) = h_u(x', y')]$$

As recommended in [2], a value to be accumulated is either hashed or encrypted before taking it to the accumulator, which hence provides strong one-wayness.

Barić and Pfitzmann in [1] noticed that for certain applications even strong one-wayness might not suffice as the adversary might be able to choose the values himself. Authors in [1] propose a stronger notion of *collision-free* accumulators and provide a construction which is secure under strong RSA assumption (Definition 5). *Collision-free accumulators* capture the notion that a probabilistic polynomial-time adversary can not generate a set of values $Y' = \{y_1, y_2, \ldots, y_n\}$ yielding an accumulated value z for which she can generate another value $y' \notin Y'$ and a witness w' such that $f(w', y') = z$.

Definition 5 (Strong RSA assumption). *When the modulus N is sufficiently large and randomly generated, and a value z is given, it is hard to find x and y satisfying $x^y \mod N = z$.*

The authors further generalize the definition of accumulators to include accumulators that do not require the quasi-commutative property. Following the terminology of [1], the class of accumulators as defined in [2] are referred to as *elementary accumulators*. Moreover, as proved in [1], collision-resistance under strong RSA assumption is achieved only when the values to be accumulated are prime.

As raised by Benaloh and de Mare, one should be able to construct accumulators without trapdoor. Trapdoors are redundant in an accumulator scheme. The party who provides N during the system setup, knows also the trapdoor p and q. Unfortunately, a party who knows p and q can completely bypass the system's security. Because, by knowing p and q, it is possible to recover the initial value, and then independently accumulate extra values and generate false witnesses. A non-trapdoor solution would not rely on trusted on-line or off-line services. A non-trapdoor accumulator is presented in [25] and is provably secure in the standard model. The authors suggest to use *generalized RSA modulus **Of Unknown complete Factorization*** and refer it as RSA-UFOs. A number N is a RSA-UFO if N has at least two large prime factors p and q such that it is infeasible for any coalition of players including those that generated N to find a splitting of N into factors N_1 and N_2 such that $p|N_1$ and $q|N_2$. A probabilistic algorithm is also proposed to generate such numbers. The security is proved in the standard model under a new assumption called *Strong RSA-UFO Assumption*. This assumption is very similar to strong RSA assumption (Definition 5), the only difference being that the modulus N is set to be a RSA-UFO.

Unlike the previous works [1, 2, 22, 25], where the list of values to be accumulated is statically defined, Camenisch *et al.* in [10] introduce the notion of *dynamic accumulators*. A dynamic accumulator permits to dynamically add and delete values to or from the accumulator, such that the cost of an add or delete is independent of the number of accumulated values. The authors use the standard RSA function based construction as proposed in [1]. To achieve constant cost of updating the witnesses the factorization of the modulus is used as a trapdoor. The security of the scheme relies on strong RSA assumption.

Nguyen in [21] proposes a dynamic accumulator based on bilinear pairings. The author considers a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (the construction also holds in the asymmetric case). The groups have the same order p and $\mathbb{G}$ is generated by P. The values to be accumulated need not to be prime but should only belong to the finite field $\mathbb{Z}/p\mathbb{Z}$. An initial seed $x \in \mathbb{Z}/p\mathbb{Z}$ and a trapdoor $s \in \mathbb{Z}/p\mathbb{Z} \setminus \{0\}$ are chosen to be used in the quasi-commutative function $f : \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ where $f(x, y) = x \cdot (y + s)$. A function $g : \mathbb{Z}/p\mathbb{Z} \to \mathbb{G}$ where $g(v) = vP$ is used to map the output of f to an element of $\mathbb{G}$. The accumulated value z for a set $Y' = \{y_1, y_2, \ldots, y_n\} \subset \mathbb{Z}/p\mathbb{Z} \setminus \{-s\}$ is $z = g(f(x, Y')) = (x \cdot \prod_{i=1}^{n} (y_i + s))\, P$, and can be easily computed given access to $(P, sP, \ldots, s^q P)$, where q is the upper bound on the number of elements to be

accumulated. Witness w_i corresponding to a value y_i is the accumulated value for $Y' \setminus y_i$. Verification involves checking if $w_i \cdot (y_i + s) = z$ which can be done using the pairing e by verifying if:

$$e((y_i + s)P, w_i) = e(z, P)$$

The security of the scheme is based on the *q-Strong Diffie-Hellman Assumption.*

Definition 6 (q-Strong Diffie-Hellman assumption [6]). *Given a tuple $t = (p, \mathbb{G}, P)$, where p is prime, $\mathbb{G}$ is a cyclic group generated by P and a tuple of values in $\mathbb{Z}/p\mathbb{Z}$ of the form $(P, sP, \ldots, s^qP)$, where $s \in \mathbb{Z}/p\mathbb{Z} \setminus \{0\}$, for any probabilistic polynomial-time algorithm $\mathcal{A}$ the following probability is negligible:*

$$Pr[A(t, P, sP, \ldots, s^qP) = (c, \frac{1}{s+c}P)) \wedge (c \in \mathbb{Z}/p\mathbb{Z})]$$

Authors in [27] provide an attack against collision-resistance property of the scheme and hence refute the claim in [21] that the accumulator is collision-resistant. The attack is based on the improperly defined security model where the adversary has access to the functions f and g. The proposed patch consists of providing the composite function $g(f(\cdot))$ to the adversary instead of the functions f and g separately. The patch proposed by the author however does not prevent other kinds of attacks and it has been proven that the scheme is insecure [32]. Another dynamic pairing based accumulator is proposed in [9], which provides a more efficient witness update algorithm.

In their survey on accumulators [12], Fazio and Nicolosi noted that the construction in [10] is such that the time to update a witness after m changes to the accumulator is proportional to m. They posed the question whether *batch update* was possible, namely if it was possible to build an accumulator where the time to update the witnesses is independent of the number of changes to the accumulated set. Wang et al. answered in affirmative by designing an accumulator with batch update in [29] and later improved in [28]. The scheme is based on Paillier cryptosystem [23] and was proved to be secure under a new assumption called *extended strong RSA assumption*, which is a variant of strong RSA assumption (Definition 5) with modulus N^2. However, contrary to the claim, Camacho et al. in [8] exhibit an attack and further prove that the time to update a witness must be at least $\Omega(m)$ in the worst case. Hence this provides an impossibility result on batch-update capable accumulators.

The previous works only generate membership witnesses, however in certain scenarios non-membership witnesses might be inevitable. In [18] authors provide a dynamic accumulator which supports short witnesses for both membership and non-membership, which they call *universal accumulators.* The initial value of the accumulator has to be public to be able to verify the non-membership witnesses. The construction is based on RSA-function and hence only prime values are permitted to be accumulated.

Karlof *et al.* in [16] use elliptic curve to build an accumulator. To accumulate the values(scalars) they are multiplied with the public-key (*i.e.* a scalar times

the base point of the curve). Witness generation follows the same algorithm but excludes the corresponding value. Verification is simple and involves checking for equality if the multiplication of the witness and the value equals the accumulated value.

2.2.2 Symmetric Accumulators A symmetric cryptographic accumulator is a trapdoor-less construction, which does not require a witness for verification. The existing constructions are secure in the *random oracle model*. Symmetric accumulators [22, 30] fundamentally consists of a one-way function $f : Y \to X$ and a vector $\boldsymbol{x} \in X$ of length ℓ, which is initialized to $\mathbf{0}$. The set of values $\{y_1, y_2, \ldots, y_n\}$ is accumulated to a vector $\boldsymbol{z}$ as:

$$\boldsymbol{z} = \boldsymbol{x} \vee \boldsymbol{f}(y_1) \vee \boldsymbol{f}(y_2) \vee \ldots \vee \boldsymbol{f}(y_n)$$

where $\vee$ is the bitwise inclusive or. Given the accumulated vector $\boldsymbol{z}$ and a value y_i, authenticating membership in the accumulated vector consists in computing $\boldsymbol{v} = \boldsymbol{f}(y_i)$ and verifying that $\forall k \in [\![0, \ell - 1]\!]$, $\boldsymbol{v}_k = 1$ implies $\boldsymbol{z}_i = 1$. Authentication is simpler in case of symmetric accumulators for it does not require computing a witness. However they suffer from long-output of the accumulator. In fact, the length of the accumulator depends on the number of values added to the accumulator and not only on the security parameter.

Nyberg in [22] proposes a symmetric accumulator. The idea is to use a hash function to generate a hashcode for values to be accumulated. Each hashcode h is considered to be composed of r blocks $h_1, h_2 \ldots, h_r$ each of size d bits. Such a code is then mapped to an r bit string, by mapping each block to a bit. A block is mapped to 0 if it is a sequence of r 0s, else it is mapped to 1. The accumulated value z is computed as the co-ordinate-wise bit-product of the strings corresponding to the values to be accumulated. To verify the membership of a value y, one computes the corresponding bit string y' of length r and checks that, for all $1 \leq i \leq r$, whenever $y'_i = 0$, then $z_i = 0$.

Bloom filters [5] by construction can be used as an accumulator. Furthermore, Yum et al. in [30] prove that they excel over other symmetric accumulators. Secure Bloom filter consists of k hash functions $\{f_i : Y \to X\}$. The functions in fact belong to a hash family. Each hash function uniformly returns a vector index. To add a value to the accumulator, it is fed to each of the hash functions to obtain k indices. The bits of $\boldsymbol{x}$ at these indices are set to 1. To verify if a given value was accumulated, the k hash functions are again applied to obtain the vector indices. If any of the bits of the accumulated vector at these indices is 0, the value was certainly not accumulated. If all the bits at these indices are 1, then we might obtain a *false positive* response. Another variant of Bloom filter has been studied in the past, where hash functions are replaced by Hash-based Message Authentication Code (HMAC).

We note that the size ℓ in case of symmetric accumulators increases with the number of elements in the filter or if the false positive rate is set lower.

3 Experiments

Our aim is to evaluate the performances of existing cryptographic accumulators in order to compare them. For this, we have implemented the original RSA-based accumulator [1,2], ECC-based accumulator described in [16], secure Bloom filter using cryptographic hash functions [22,30] and HMAC-SHA-1 based accumulator mentioned in [13]. This work only considers static accumulators as they capture the essential notion of membership testing. Furthermore, dynamic variants differ from static accumulators only in the case when witnesses are required to be updated in case of deletion/addition of a value from/to an accumulator. For symmetric accumulators, we only consider Bloom filter for our evaluation as they perform better (see [30]) over accumulator proposed by Nyberg [22].

As discussed earlier, several pairing-based accumulators have been proposed in the past. However, due to continuous and recent attacks [15] on existing pairing friendly curves, no officially recommended curves have been proposed until now. We hence exclude pairing-based accumulators from our evaluation. However, interested readers may refer to [17, 24] where older curves have been considered for benchmarking.

In the rest of this section, we give the scenario used for our benchmarks, the chosen cryptographic parameters, the setting of our evaluations and preliminary analysis on the memory and communication costs of accumulators we have considered.

3.1 Scenario and Cryptographic Parameters

To benchmark the different cryptographic accumulators, we used different size S for the accumulated values. The size S varies from 128 to 2048 bytes. The smallest representing the acceptable RSA public-key of 1024 bits, and in general these sizes are representative of X.509 certificates. The number n of values in an accumulator is arbitrarily fixed to 1000.

In case of Bloom filter, if ϵ is the probability of false positive, then the number of hash functions used is $k = -\log \epsilon$.

The respective parameters of each accumulator and the list of curves used in our experiments (integrated in OpenSSL) are given in Table 1.

Table 1. Security and parameters used in our experiments.

Accumulator	**Parameters size**		
RSA	1024	2048	3072
ECC binary	163	283	571
ECC prime	160	256	521
Bloom ($-\log_2 \epsilon$)	128	256	512

Prime field	**Binary field**
secp160r1	sect163r1
secp256r1	sect283r1
secp521r1	sect571r1

3.2 Software and Settings

All implementations are in C. RSA accumulator is implemented using GNU multi-precision library GMP[3] version 4.2.1 to handle arbitrary precision arithmetic on integers. ECC-based accumulator uses OpenSSL (version 1.0.1) EC library.

To ensure strong one-wayness and collision resistance required for accumulators [2], we use SHA-1, HMAC-SHA-1, SHA-256 and SHA-3 [3]. The values to be accumulated are hashed using one of these functions before adding it to the accumulator. The OpenSSL implementation of SHA-1, HMAC-SHA-1 and SHA-256 has been used. The optimized code of Keccak available in the version 3.2 [4] for SHA-3 as well as a naive implementation have been used.

Our target platform is a 64-bit processor desktop computer powered by an Intel i7 3520M processor at 2.90 GHz with 4 MB cache and running 3.8.0-35 Ubuntu Linux. We have used GCC 4.3.3 with `-O3` optimization flag.

3.3 Memory and Communication Cost

In applications, memory and communication costs are critical. The size of the accumulator is an inevitable memory cost. The size ℓ of a secure Bloom filter is given by $\ell = \frac{-n \log_2(\epsilon)}{\ln 2}$ with ϵ the false positive probability, see [7]. For $\epsilon = 2^{-128}$ and $n = 1000$, a secure Bloom filter (184,664 bits) is 180 times bigger than a 1024-bit RSA accumulator and 1154 times bigger than a 160-bit ECC accumulator. The key size for RSA and ECC determines the size of the accumulator. Therefore, ECC-based accumulators outperform RSA-based ones.

In Fig. 1, we show the evolution of ℓ (size of the accumulator) for different accumulators as a function of n. For $n \geq 12$, even 2048-bit RSA accumulators are smaller than secure Bloom filter. The 160-bit ECC accumulator is obviously less expensive than RSA based accumulator and hence it is often the best solution in terms of memory.

If secure Bloom filters are costly in memory, they are inexpensive in communication: they do not require witnesses as in ECC and RSA. The drawback of asymmetric accumulators is the need to generate and communicate the witnesses. In the next section, we investigate the speed of accumulators.

4 Results

For all accumulators, we measure the time needed to verify if a value belongs to an accumulator. The results presented in the paper are the average values for 100000 repetitions of the same experiment. We first give individual results on each accumulator and then make an overall comparison. We note that the subfigures in this section are plotted in the same scale as the one used in the leading plot of the figure.

[3] `http://gmplib.org/`
[4] `http://keccak.noekeon.org/files.html`

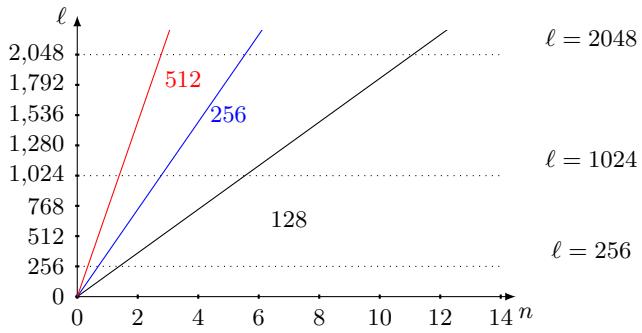

Fig. 1. Memory cost of accumulators. The cost of Bloom filters is given for different false positive rates respectively $\epsilon = 2^{-128}$ (black), $\epsilon = 2^{-256}$ (blue) and $\epsilon = 2^{-512}$ (red). The horizontal dashed line represents the cost of asymmetric accumulators (ECC and RSA).

4.1 Individual Results

4.1.1 RSA-based We use the following RSA key sizes $\{1024, 2048, 3072\}$. In Fig. 2, we first observe that the key length has (obviously!) a big influence on the performance. If we consider SHA-1 as deprecated, the choice of the hash function (SHA-256 or SHA-3) has little influence. Moreover, the message size S also has no influence on the verification time. The cost of hashing is subdued by the cost of exponentiation. We note that this accumulator only provides strong one-wayness as the values are only hashed before adding them to the accumulator.

As discussed by Barić and Pfitzmann [1] a stronger notion of security called collision-freeness is achieved when the values added to an accumulator are primes or prime representatives of non-prime values. We also measure the cost incurred by collision-free RSA accumulators using prime representation of input values. The method of computing prime representatives follows the description by Barić and Pfitzmann [1]. The prime representative $p(y)$ for a value y is generated by computing the digest $h(y)$ as in the previous technique and then by finding a t-bit number d which when appended to $h(y)$ as $p(y) = 2^t h(y) + d$ makes $p(y)$ prime. The authors suggest to use $t = \lceil \log_2(2 \times \text{size of RSA modulus}) \rceil$. The integer d then can be found with high probability by testing for primality of $p(y)$ for each odd integer d satisfying $1 \leq d < 2^t$. Table 2 presents the significant overhead incurred if one wishes to achieve collision freeness instead of strong one-wayness according to the used hash function.

	SHA-1	**SHA-256**	**SHA-3**
Overhead	×23	×85	×78

Table 2. Overhead for using a stronger security model.

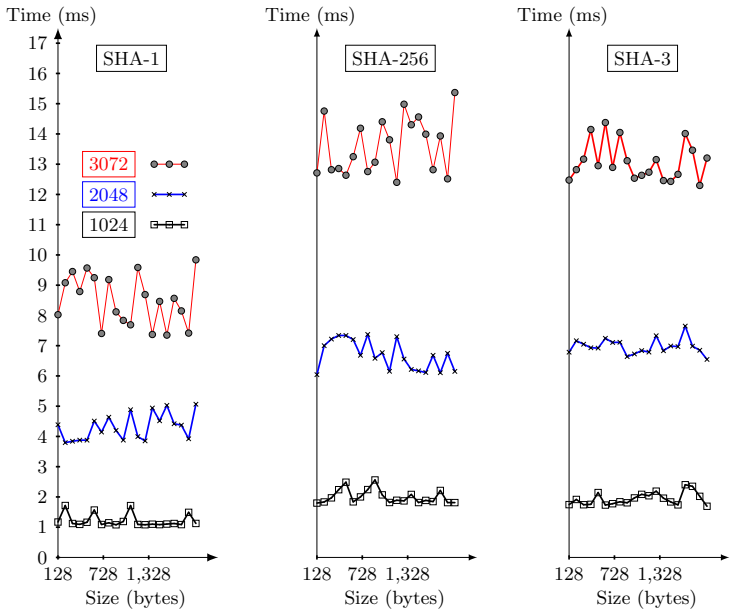

Fig. 2. Verification time for RSA-based accumulators with different key size $\{1024, 2048, 3072\}$ with SHA-1, SHA-256 and SHA-3.

4.1.2 ECC-based We observe from Fig. 3 and Fig. 4 that for almost the same security level binary elliptic curve of degree 163 performs better than elliptic curve over 160 bit prime field. The former takes around 2 μs for verifying the witness of a value of size 128 bytes while the later takes 10 μs in case of SHA-1. Hence an overhead of 8 μs, if one wishes to use prime fields. We further remark that unlike in RSA accumulator where hash function has no impact, in case of elliptic curve, the impact of hash functions (SHA-1 and SHA-256) appears visible for larger size of the values. This phenomenon appears because for large values cost of hashing is comparable to the cost of point multiplication on elliptic curve. Unlike other hash functions, SHA-3 does not influence the verification time as we use the optimized version of Keccak.

4.1.3 Symmetric Accumulators In Fig. 5, we analyze the verification time for different size of values for SHA-1, HMAC-SHA-1, SHA-256 and two implementations of SHA-3 (a naive one and the optimized code) with $\epsilon \in \{2^{-128}, 2^{-256}, 2^{-512}\}$. If k is the number of calls to the cryptographic hash function. We have $k = -\log_2(\epsilon)$ (see [7]). Any effect or artifact on the performance of the hash function is amplified k times on the verification using a secure Bloom filter. It explains why the verification time grows linearly for SHA-1, HMAC-SHA-1, SHA-256 and the naive implementation of SHA-3. For the optimized implementation of SHA-3, we observe a step-wise progression. The size of the step is 1000 bytes.

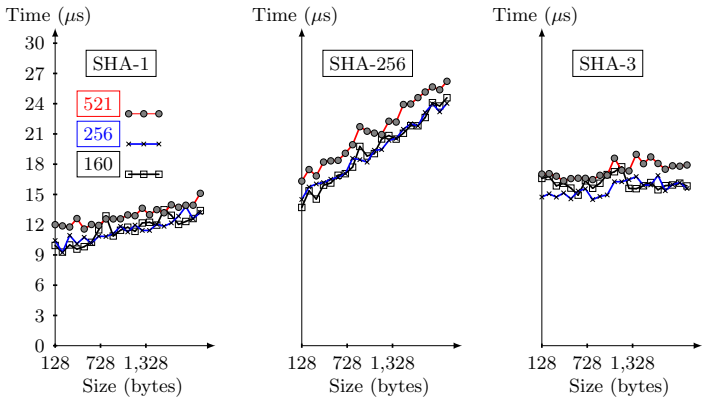

Fig. 3. ECC prime

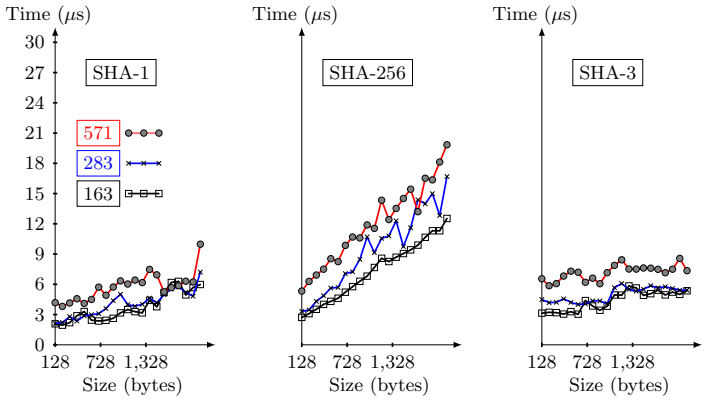

Fig. 4. ECC binary

4.2 Overall Comparison

In Figure 6, we compare all accumulators for a security level of 128 bits and a value of 128 bytes using SHA-1 and SHA-3. Each time, ECC-based accumulators are the most efficient and hence the best of all, then we have Bloom filter and at the bottom the RSA-based accumulators.

We note that building a RSA-based accumulator with n elements would cost n-times the cost of a verification. This is due to the fact that building an accumulator would involve n modular exponentiations. The same holds for ECC-based accumulator, where the point multiplication is computed n times. Bloom filter too behaves in the same manner, where hash would be computed for all the n values. However, as Bloom filter is witness free, cost of adding a new value would be the cost of computing the k hashes corresponding to the new value *i.e.* cost of a verification. While, in case of non-dynamic RSA and ECC-based

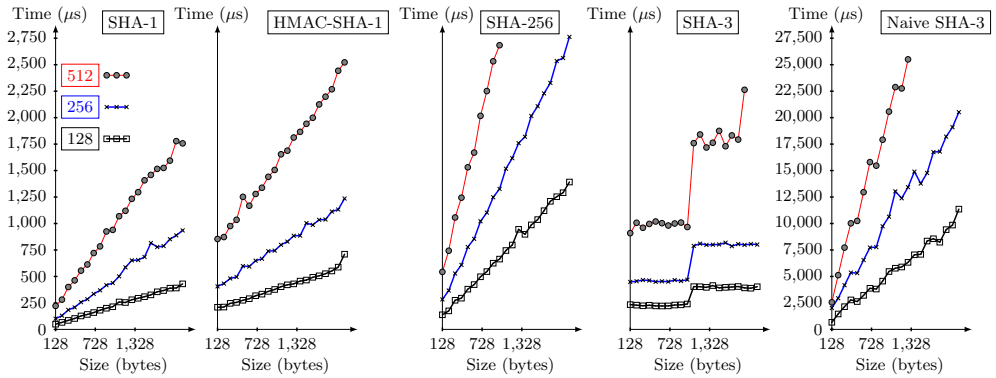

Fig. 5. Verification time for secure Bloom filter as a function of the value's size and for different false positive values $\epsilon \in \{2^{-128}, 2^{-256}, 2^{-512}\}$.

accumulators, the cost of adding a new value would entail updating each of the previous n witnesses. A representative result for build and add time is presented in Table 3.

Table 3. Time to build accumulator (for 1000 values) and add a new value using SHA-1 for 128 bytes' values.

		Time (μs)		
Accumulator	**Param.**	**Verif.**	**Build**	**Add**
ECC binary	163	2	2000	2200
ECC prime	160	10	10000	10100
Bloom ($-\log_2 \epsilon$)	128	49	49000	49
RSA	1024	1100	1100000	1102000

5 Related Work

The most potentially relevant work on performance of cryptographic accumulators is presented by Papamanthou *et al.* [24]. The authors first use accumulators to design *authenticated hash tables*, where a user queries a hash table held by a server and wishes to verify the correctness of the response received. The authors then benchmark the authenticated hash table scheme using RSA accumulator and pairing based accumulator. In practice, RSA accumulator based scheme performs better than the pairing based accumulator scheme. However, RSA accumulator has a logarithmic advantage over pairing in the asymptotic case. We note that the evaluation is restricted to 1024 bits RSA modulus and Type A pairings. Furthermore, the values to be accumulated are generated using arbitrarily chosen SHA-256. In a later work, Lapon *et al.* [17] benchmark accumulator based

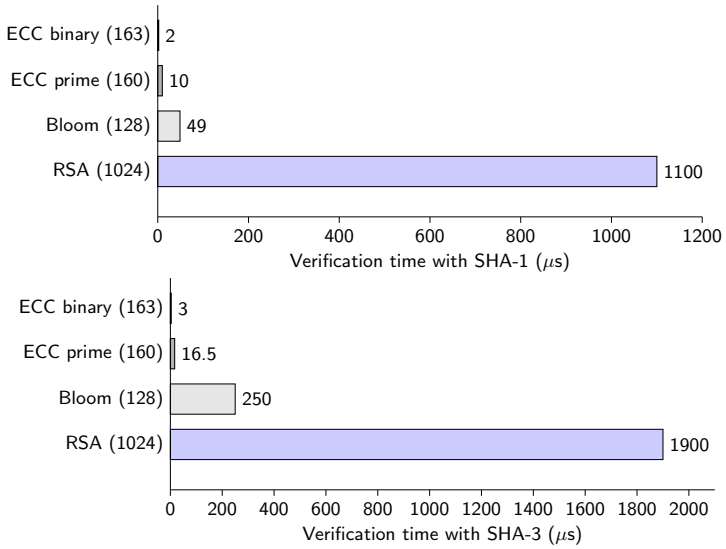

Fig. 6. Summary of different accumulators for 128 bytes' values.

credential revocation schemes. However, only pairing-based schemes are considered for performance evaluation. Another relevant work is an extensive study of the real-world performance of authenticated dictionary schemes by Crosby and Wallach [11]. The authors use an authenticated dictionary based on RSA accumulators as one of their test systems, and conclude that the RSA accumulators introduce a significant performance overhead that makes this authenticated dictionary too slow for practical use compared to digital signatures. However, the results do not show the specific amount of time taken by RSA accumulator operations because they are formatted in terms of authenticated dictionary operations such as inserts and updates.

The previous works in general only consider certain accumulators coupled with an arbitrarily chosen hash function and fixed security parameters in certain cases. In this perspective, our work fills this gap by providing a complete evaluation of different schemes using varied security levels, achieving different security notions and studying impact of other implicit parameters such as hash functions.

6 Conclusion

After a survey of existing cryptographic accumulators, we evaluate their merits. Our work considers rather big-size data ranging from 128 bytes (1024 bits) to 2048 bytes. These sizes are appropriate for Certificate Revocation Lists (CRL) which are widely used in PKI infrastructure. It is important to notice that

CRL file size can become very large, for example CRLs issued by VeriSign[5] can be a megabyte in size. In order to efficiently store such sensitive data non-cryptographic Bloom filters are often used as a space efficient data structure. Our analysis demonstrates that for such application scenarios ECC can reduce the memory footprint and improve the verification at the cost of a witness.

References

1. N. Bari and B. Pfitzmann, "Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees," in *Advances in Cryptology - EUROCRYPT '97*. Springer, 1997.
2. J. C. Benaloh and M. de Mare, "One-Way Accumulators: A Decentralized Alternative to Digital Sinatures," in *Advances in Cryptology - EUROCRYPT '93*. Springer, 1993.
3. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The Keccak reference," 2011, http://keccak.noekeon.org/.
4. J. Bethencourt, D. X. Song, and B. Waters, "New Techniques for Private Stream Searching," *ACM Transactions on Information and System Security*, vol. 12, no. 3, 2009.
5. B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, 1970.
6. D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," in *Advances in Cryptology CRYPTO 2004*, ser. Lecture Notes in Computer Science, M. Franklin, Ed. Springer Berlin Heidelberg, 2004, vol. 3152, pp. 41–55.
7. A. Z. Broder and M. Mitzenmacher, "Survey: Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, 2003.
8. P. Camacho and A. Hevia, "On the Impossibility of Batch Update for Cryptographic Accumulators," in *Progress in Cryptology LATINCRYPT 2010*, ser. Lecture Notes in Computer Science, vol. 6212. Springer Berlin Heidelberg, 2010, pp. 178–188.
9. J. Camenisch, M. Kohlweiss, and C. Soriente, "An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials," in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, ser. Irvine. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 481–500.
10. J. Camenischn and A. Lysyanskaya, "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials," in *Proceedings of the 22Nd Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '02. London, UK, UK: Springer-Verlag, 2002, pp. 61–76.
11. S. A. Crosby and D. S. Wallach, "Authenticated dictionaries: Real-world costs and trade-offs," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, 2011.
12. N. Fazio and A. Nicolosi, "Cryptographic Accumulators : Defintions, Constructions and Applications," Technical Report, 2002.
13. E.-J. Goh, "Secure Indexes," Cryptology ePrint Archive, Report 2003/216, 2003, http://eprint.iacr.org/2003/216/.
14. J. Haas, Y.-C. Hu, and K. Laberteaux, "Efficient certificate revocation list organization and distribution," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 3, pp. 595–604, 2011.

[5] `http://crl.verisign.com`

15. T. Hayashi, T. Shimoyama, N. Shinohara, and T. Takagi, "Breaking pairing-based cryptosystems using t pairing over gf(397)," in *Advances in Cryptology - ASIACRYPT 2012*. Springer, 2012.
16. C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. D. Tygar, "Distillation Codes and Applications to DoS Resistant Multicast Authentication," in *Network and Distributed System Security Symposium - NDSS 2004*. The Internet Society, 2004.
17. J. Lapon, M. Kohlweiss, B. Decker, and V. Naessens, "Performance analysis of accumulator-based revocation mechanisms," in *Security and Privacy Silver Linings in the Cloud*. Springer Berlin Heidelberg, 2010.
18. J. Li, N. Li, and R. Xue, "Universal Accumulators with Efficient Nonmembership Proofs," in *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*, ser. ACNS '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 253–269.
19. Z. Li, "Efficient Authentication, Node Clone Detection, and Secure Data Aggregation for Sensor Networks," Ph.D. dissertation, University of Waterloo, 2010.
20. H. Massias, "La certification cryptographique du temps," Ph.D. dissertation, Université catholique de Louvain, 2000.
21. L. Nguyen, "Accumulators from Bilinear Pairings and Applications," in *Proceedings of the 2005 International Conference on Topics in Cryptology*, ser. CT-RSA'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 275–292.
22. K. Nyberg, "Fast Accumulated Hashing," in *Fast Software Encryption - FSE 1996*. Springer, 1996.
23. P. Paillier, "Public-Key cryptosystem based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99*, ser. Lecture Notes in Computer Science 1592. Springer, 1999, pp. 223–238.
24. C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Authenticated hash tables," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*. ACM, 2008.
25. T. Sander, "Efficient Accumulators Without Trapdoor Extended Abstracts," in *Proceedings of the Second International Conference on Information and Communication Security*, ser. ICICS '99. London, UK, UK: Springer-Verlag, 1999, pp. 252–262.
26. T. Sander, A. Ta-Shma, and M. Yung, "Blind, auditable membership proofs," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, Y. Frankel, Ed. Springer Berlin Heidelberg, 2001, vol. 1962, pp. 53–71.
27. C. Tartary, S. Zhou, D. Lin, H. Wang, and J. Pieprzyk, "Analysis of bilinear pairing-based accumulator for identity escrowing," *IET Information Security*, vol. 2, no. 4, pp. 99–107, 2008.
28. P. Wang, H. Wang, and J. Pieprzyk, "Improvement of a Dynamic Accumulator at ICICS 07 and Its Application in Multi-user Keyword-Based Retrieval on Encrypted Data," in *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, 2008, pp. 1381–1386.
29. ——, "A New Dynamic Accumulator for Batch Updates," in *Proceedings of the 9th International Conference on Information and Communications Security*, ser. ICICS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 98–112.
30. D. H. Yum, J. W. Seo, and P. J. Lee, "Generalized Combinatoric Accumulator," *IEICE Transactions*, vol. 91-D, 2008.
31. J. Zachary, "A decentralized approach to secure management of nodes in distributed sensor networks," in *Military Communications Conference, 2003. MILCOM '03. 2003 IEEE*, vol. 1, 2003.

32. F. Zhang and X. Chen, "Cryptanalysis and Improvement of an ID-based Ad-hoc Anonymous Identification Scheme at CT-RSA 05," *Inf. Process. Lett.*, vol. 109, no. 15, pp. 846–849, 2009.