Performances of Cryptographic Accumulators

Amrit Kumar
Privatics team
INRIA Rhône-Alpes
Grenoble, France
Email: amrit.kumar@inria.fr

Pascal Lafourcade
LIMOS
Université d'Auvergne
Clermont Ferrand, France
Email: pascal.lafourcade@imag.fr

Cédric Lauradoux
Privatics team
INRIA Rhône-Alpes
Grenoble, France
Email: cedric.lauradoux@inria.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 thorough 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.**

***Index Terms*—Cryptographic accumulators, Bloom filter, Performance**

I. Introduction

Cryptographic accumulators are space/time efficient data structures that are used to test if a value belongs to a given set. They are the cryptographic counterpart of a data structure very popular in the field of networking: the Bloom filter [3].

Similar to a one-way hash function, cryptographic accumulators generate a fixed-size digest representing an arbitrarily large set of values. Interestingly, an *asymmetric accumulator* further provides a fixed-size *witness* for any value of the set, which can be used together with the accumulated digest to verify its membership in the set. The security requirement is that the verification succeeds only for the elements of the set.

Since their appearance as a cryptographic primitive in [2], 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 search on encrypted data [5] to data aggregation in sensor networks [10] and distillation codes [7] among several others.

Despite their widespread applications, 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, Zachary [14] 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.

Some of the previous works ([4], [9], [12]) *partially* study the performance of accumulator based primitives. These in general only consider certain accumulators coupled with an arbitrarily chosen hash function and fixed security parameters in certain cases. In this perspective, we fill this gap by providing a *complete evaluation* of different schemes (*symmetric* as well as asymmetric) in a *stand-alone* manner, using varied security levels and studying the impact of other implicit parameters such as hash functions.

Contributions: We provide a software performance evaluation (see [8] for the full version of this paper) 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.

We first give cryptographic recalls concerning accumulators and provide a few asymmetric and symmetric constructions.

II. Cryptographic Accumulators

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.

Definitions: 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* (Def. 1) which satisfy an additional *quasi-commutative* property (Def. 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 for $1 \leq i \leq n$:

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

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. The accumulator is secure if the verification fails for any $y \in Y \setminus Y'$ with a high probability.

A. Some Instances

We briefly present below the accumulators we study in this work, which can be broadly classified into: asymmetric and symmetric ones.

1) Asymmetric Accumulators: Asymmetric accumulators are those which require witness for verification and are built on asymmetric cryptographic primitives. The first accumulator proposed by Benaloh and de Mare [2] is asymmetric. 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*, Def. 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 [11], 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* (Def. 4) where the attacker is not imposed the choice of y' as in Def. 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:*

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

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[1].

Another construction proposed by Karlof *et al.* [7] uses 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) Symmetric Accumulators: Symmetric accumulators are those which do not require witness for verification and are built on symmetric cryptographic primitives. Bloom filters [3] by construction can be used as a symmetric accumulator. Furthermore, Yum et al. in [13] prove that they excel over other symmetric accumulators. Secure Bloom filter consists of k hash functions $\{f_i : Y \to X\}$ and a vector $\vec{x}$ (of size ℓ) initialized to $\vec{0}$. 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 $\vec{x}$ 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, then value was certainly not accumulated. If all the bits at these indices are 1, then the value is in the filter (with a small *false positive* probability). Another variant of secure Bloom filter uses Hash-based Message Authentication Code (HMAC) instead of hash functions.

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

III. 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 [7], secure Bloom filter using cryptographic hash functions [13] and HMAC-SHA-1 based accumulator mentioned in [5].

We note that another variant of cryptographic accumulator called *dynamic* accumulators exist, which allow constant-time addition/deletion to/from the accumulator. However, this work only considers *static* accumulators (as previously presented), since they capture the essential notion of membership testing.

[1]Another stronger notion of *collision resistance* (requiring the values to be prime) has also been studied [1], but due to space constrainsts, it has not been included in this work. See the full version for details and experiments [8].

For symmetric accumulators, we only consider Bloom filter for our evaluation as they perform better (see [13]) over accumulator proposed by Nyberg [11].

Several *pairing-based* accumulators have also been proposed in the past. However, due to continuous and recent attacks [6] 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 [9], [12] where older curves have been considered for benchmarking.

A. 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 (minimum acceptable RSA key size) to 2048 bytes which is 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 (optimal) number of hash functions considered is $k = -\log_2(\epsilon)$.

The respective parameters of each accumulator and the elliptic curves (integrated in OpenSSL) are given in Table I.

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

TABLE I
SECURITY AND PARAMETER SIZES IN OUR EXPERIMENTS.

B. Software and Settings

All implementations are in C. RSA accumulator is implemented using GNU multi-precision library GMP[2] 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 required for accumulators [2], we use SHA-1, HMAC-SHA-1, SHA-256 and SHA-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 version 3.2 [3] has been used for SHA-3.

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.

C. 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. 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 respective accumulator. Therefore, ECC-based accumulators outperform RSA-based ones.

For $\epsilon = 2^{-128}$ and $n \geq 12$, 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. In the next section, we investigate the speed of accumulators.

IV. 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.

A. Individual Results

1) RSA-based: We use the following RSA key sizes $\{1024, 2048, 3072\}$. In Fig. 1, 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.

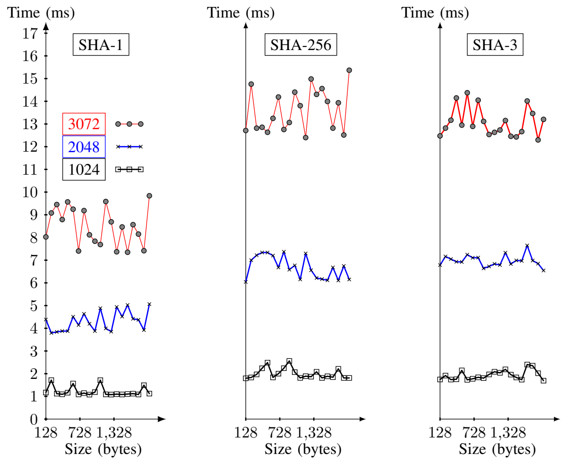

Fig. 1. Verification time for RSA-based accumulators with different key size $\{1024, 2048, 3072\}$ with SHA-1, SHA-256 and SHA-3.

2) ECC-based: We observe from Fig. 2 and Fig. 3 that for almost the same security level binary elliptic curve of degree 163 performs better than elliptic curve over 160 bit prime field. 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.

[2]http://gmplib.org/

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

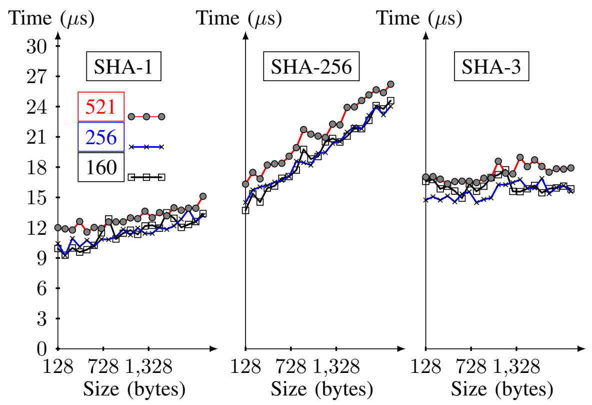

Fig. 2. ECC prime

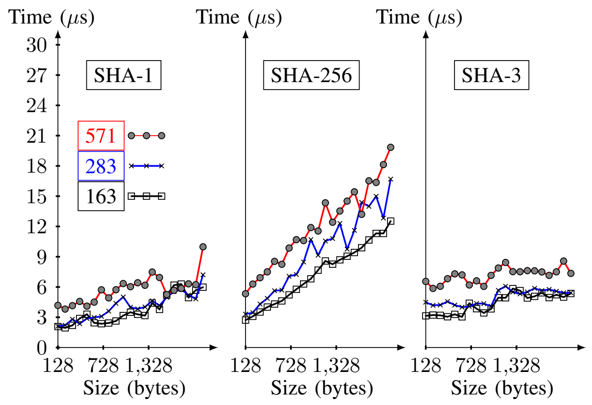

Fig. 3. ECC binary

3) Symmetric Accumulators: In Fig. 4, we analyze the verification time for different size of values for SHA-1, HMAC-SHA-1, SHA-256 and SHA-3 with $\epsilon \in \{2^{-128}, 2^{-256}, 2^{-512}\}$. If k is the number of calls to the cryptographic hash function. We have $k = -\log_2(\epsilon)$. 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 and SHA-256. For optimized SHA-3, we observe a step-wise progression. The size of the step is 1000 bytes.

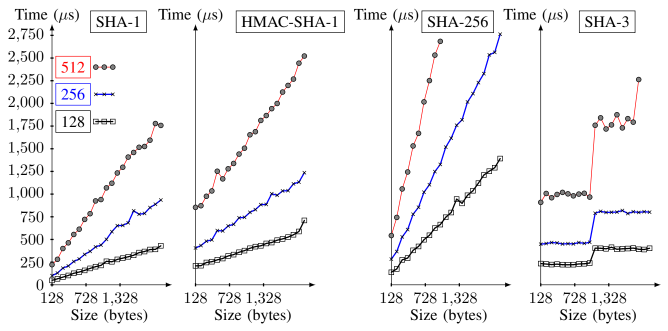

Fig. 4. Verification time for secure Bloom filter as a function of the value's size and for different false positive values $\epsilon \in \{2^{-128}, 2^{-256}, 2^{-512}\}$.

Overall comparison: With the obtained results, we highlight that: **ECC-based accumulators are the most efficient, 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 and Bloom filter.

V. Conclusion

We present a software based performance analysis of cryptographic accumulators. 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[4] 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.

Acknowledgement. This research was conducted with the support of the "Digital trust" Chair from the University of Auvergne Foundation. The work was also partly supported by the LabEx PERSYVAL-Lab (ANR–11-LABX-0025) and the project-team SCCyPhy.

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] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, 1970.

[4] 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.

[5] E.-J. Goh, "Secure Indexes," Cryptology ePrint Archive, Report 2003/216, 2003, http://eprint.iacr.org/2003/216/.

[6] 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.

[7] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. D. Tygar, "Distillation Codes and Applications to DoS Resistant Multicast Authentication," in *NDSS*. The Internet Society, 2004.

[8] A. Kumar, P. Lafourcade, and C. Lauradoux, "Performances of Cryptographic Accumulators," Tech. Rep., 2014, http://hal.archives-ouvertes.fr/hal-00999432.

[9] 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.

[10] Z. Li, "Efficient Authentication, Node Clone Detection, and Secure Data Aggregation for Sensor Networks," Ph.D. dissertation, University of Waterloo, 2010.

[11] K. Nyberg, "Fast Accumulated Hashing," in *Fast Software Encryption - FSE 1996*. Springer, 1996.

[12] C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Authenticated hash tables," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*. ACM, 2008.

[13] D. H. Yum, J. W. Seo, and P. J. Lee, "Generalized Combinatoric Accumulator," *IEICE Transactions*, vol. 91-D, 2008.

[14] 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.

[4]http://crl.verisign.com