A Security Protocol for Self-Organizing Data Storage

Nouha Oualha, Melek Önen and Yves Roudier

Abstract This paper describes a cryptographic protocol for securing self-organized data storage through periodic verifications. The proposed verification protocol, which goes beyond simple integrity checks and proves data conservation, is deterministic, efficient, and scalable. The security of this scheme relies both on the ECDLP intractability assumption and on the difficulty of finding the order of some specific elliptic curve over $\mathbb{Z}_n$. The protocol also makes it possible to personalize replicas and to delegate verification without revealing any secret information.

1 Introduction

Online data storage has become an increasingly popular and important application, especially given the increasingly nomadic use of data and the ubiquity of data producing processes. As illustrated by P2P infrastructures like AllMyData, Wuala, or Ubistorage, self-organization today represents a promising approach to achieving scalable and fault-tolerant storage, even though it proves far more demanding in terms of security than plain distributed storage. This paper considers such a self-organizing storage application in which a peer, the data *owner*, replicates its data by storing them at *holder* peers.

Ensuring the availability of stored data in particular requires periodic verifications of the remote storage at peers for detecting voluntary data destruction by holders, which simple integrity checks cannot achieve. Such an interactive check may be formulated as a proof of knowledge in which the holder attempts to convince the verifier that it possesses some data, which is demonstrated by correctly responding to queries that require computing on the very data. Such a verification should neither require transferring back the entire data nor make it necessary to store large data at a verifier. Some authors have emphasized the difference between this type

EURECOM, France, e-mail: {Nouha.Oualha|Melek.Onen|Yves.Roudier}@eurecom.fr

www.manaraa.com

of proof and classical proof of knowledge protocols through the use of a specific terminology: proofs of data possession for [3, 1], and proofs of retrievability for [5].

This paper introduces a secure self-organizing storage protocol for highly dynamic P2P environments, with scalability as an essential objective. It notably makes it possible to generate an unlimited number of verification challenges from the same small-sized security metadata and is the first, to our knowledge, to introduce the secure delegation of data storage verification. This enables verification, and not only storage, to be distributed, thereby balancing verification costs among several peers while suppressing a single point of failure. It aims at the following objectives: (1) Remote detection of data destruction. (2) Collusion-resistance, in particular to selfish holders trying to optimize their resources. (3) Denial-of-Service prevention, in particular to prevent storage disruption through *flooding* holders with bogus verification requests, or the malicious *replay* of a valid challenge or response message.

2 Secure Storage Scheme

This section describes our three-party secure storage protocol. The protocol relies on the hardness of two different problems in the context of elliptic curve cryptography. The first problem is the elliptic curve discrete logarithm problem (ECDLP) which is to find r given two elements P an element of a finite field G and $Q = rP$. The second problem is related to the order of an elliptic curve, which has been proved to be computationally equivalent to factoring the corresponding composite number for some set of elliptic curves [6].

The scheme consists in four phases of Setup, Storage, Delegation, and Verification executed between an owner, a holder, and a verifier. The owner communicates the data to the holder at the storage phase and the meta-information to the verifier at the delegation phase. At the verification phase, the verifier interactively checks the holder's possession of data, which can be executed an unlimited number of times. In the following, we assume that the data is uniquely mapped into a number $d \in \mathbb{N}$ (e.g., conversion from a binary to a decimal representation). The secure storage scheme is described in Figure 1, and relies on the following polynomial time algorithms:

- `Setup`: The algorithm is run by the owner at the *setup* phase. Given a chosen security factor k ($k > 512$ bits), the algorithm outputs the parameters for generating an elliptic curve whose order N_n is hard to compute as previously explained. N_n is thus kept secret by the owner.

- `Personalize`: This algorithm prevents collusion between holders. It is run by the owner at the *setup* phase. It takes in input data d and a secret random number s. It returns d' the encryption of d with a keyed pseudo-random function such as AES.

- `MetaGen`: The algorithm is run by the owner at the *delegation* phase. It takes in input d' and returns $T = (d' mod N_n)P$. T is stored by the verifier.

www.manaraa.com

- `Challenge`: The algorithm is run by the verifier at the *verification* phase. It takes in input a random number r and returns point $Q = rP$. Q will be sent to the holder as a challenge.

- `Response`: The algorithm is run by the holder at the *verification* phase. It takes in input a point Q and an integer d' and outputs $R = d'Q$. R is sent to the verifier as a response to a challenge.

- `Check`: The algorithm takes in input the response R, the random number r of the challenge, and the metadata T. Checking if $R = rT$ decides on the holder's proof acceptance or rejection.

An improved version of this protocol whereby the computational complexity at the holder can be reduced by splitting data into m chunks is described in more detail in [8] together with solutions to DoS issues. Such an extension also makes it possible to consider our scheme in a probabilistic setting similar to [1].

Phases	Description		
Setup	**Owner** (public = (n, b, P), secret = N_n) ← `Setup`(k)		
Storage	**Owner** d' ← `Personalize`(d, s)	d' →	**Holder** store d'
Delegation	**Owner** T ← `MetaGen`(d', n, b, P)	T →	**Verifier** store T
Verification	**Verifier** Q ← `Challenge`(r, n, b, P) {"*accept*", "*reject*"} ← `Check`(R, r, T, n, b)	Q → ← R	**Holder** R ← `Response`(d', Q, c, b)

Fig. 1 Secure Storage Verification Protocol

Security analysis. This section essentially discusses completeness and soundness of the protocol, the two essential properties of a proof of knowledge protocol [4]. An extended version of the protocol [8] addresses other security attacks.

Theorem 1. *The proposed protocol is* ***complete****: if the verifier and the holder correctly follow the proposed protocol, the verifier always accepts the proof as valid; and* ***sound****: if the claimant does not store the data, the verifier will not accept the proof as valid.*

Proof. Thanks to the commutative property of point multiplication in an elliptic curve, we have $d'rP = rd'P$. Therefore, since $d'Q = rT$, the proposed protocol is *complete*. Furthermore, there are only three ways to generate a correct response without storing the data. The first one is to store $d'P$ (which is much smaller that the full data size) instead of d. In this case, the holder would have to find r which is equivalent to solving ECDLP. Another option for the holder is to compute N_n, the order of the elliptic curve, in order to store $\{d' mod N_n\}$ instead of d'. However, this is hard in our particular setting as explained above. The last option for the holder is to collude with other holders storing the data. This option cannot be considered either since data at each holder is personalized and the only peer that knows the secret s used for personalization is the owner. Therefore the proposed protocol is *sound*.

www.manaraa.com

3 Related Work

Deterministic Verification. Deterministic solutions allow verifying the remote storage of the full data at a holder through a single operation. The use of precomputed but limited series of time-variant challenges stored at the verifier has been suggested early on: [2] describes a time-variant MAC. Approaches with an unlimited number of challenges instead rely on the use of homomorphisms. In the SEC scheme [4], tags are stored with data chunks and a subset of these are combined with chunks using a discrete logarithm based homomorphic scheme. Thanks to his knowledge of the secret used to make up these tags, only the verifier can directly check the proof. [2] describes a technique, later rediscovered [3], that makes use of an RSA-based hash function H: the prover hashes the combination of a nonce sent by the verifier with the data to prove it still holds them. The verifier stores $H(data)$ and the RSA public key as a secret key: he can compute his own proof using RSA homomorphic properties and compare it with the prover's. The whole data is however used as an exponent, which is computationally intensive for the prover. [12] addresses this concern at the expense of additional storage at the verifier, the data being split into m chunks.

Probabilistic Verification. Probabilistic verification methods rely on the verification of randomly sampled stored data. They have been favored in many proposals to lessen the performance impact of verification on holders. In a first type of schemes, the verifier compares the value of a stored chunk with the value of a reference data chunk. The probability of detecting selfish holders increases with the number of chunks verified at the expense of linearly increasing communication costs. [10] proposes such a scheme which improves on [7] and the Merkle-based solution by Wagner mentioned in [4], by transferring the role of protecting reference data from the verifier to the prover using signatures. The POR protocol [5] is based on verification of random values signed and hidden within the data. The verification is probabilistic with the number of verification operations allowed being limited to the number of sentinels. Another probabilistic approach proposed in [11] makes use of Rabinesque algebraic signatures of data blocks stored at different holders, and on the homomorphic properties of the signatures with respect to parity. This approach however makes it difficult to recognize a faulty holder if the parity blocks do not match. The PDP model [1], which combines a certain number of randomly selected homomorphic verifiable tags compressed into one result far smaller in size than that of the tags, seems one of the most promising of recent schemes proposed.

4 Conclusion

This paper introduces a protocol that satisfies the security needs of self-organizing storage applications, in particular through the introduction of delegated verifications, and which also meets their performance requirements. The scheme security relies on an elliptic curve cryptographic scheme in which each challenge-response

www.manaraa.com

message mainly consists of an en elliptic curve point on $\mathbb{Z}_n$. The size of messages between the verifier and the prover is independent from the size of data and only a function of the security parameter k of the `Setup` algorithm: a smaller number of resources may be used at the expense of a reduction in the security of our scheme however. The verifier needs to store only one elliptic curve point to produce challenges at will. Finally, the construction and verification of the proof rely on point multiplication operations only. We are actively investigating the use of this protocol as an observation primitive to both stimulate peer cooperation [9] and to evolve active replication strategies to rejuvenate the replicas of some data under attack.

Acknowledgements The work presented here has been partially funded by Institut Telecom Initiative program on autonomic and spontaneous networks, projects PACALAB and SPREADS (ANR), and the ReSIST IST-026764 NoE.

References

1. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, pp. 598–609. ACM, New York, NY, USA (2007)
2. Deswarte, Y., Quisquater, J.J., Saidane, A.: Remote integrity checking. In: Conference on Integrity and Internal Control in Information Systems '03 (2003)
3. Filho, D.L.G., Barreto, P.S.L.M.: Demonstrating data possession and uncheatable data transfer. Cryptology ePrint Archive, Report 2006/150 (2006). http://eprint.iacr.org/
4. Golle, P., Jarecki, S., Mironov, I.: Cryptographic primitives enforcing communication and storage complexity. In: Financial Crypto (2002)
5. Juels, A., Burton S. Kaliski, J.: PORs: proofs of retrievability for large files. In: CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, pp. 584–597. ACM, New York, NY, USA (2007)
6. Koyama, K., Maurer, U., Okamoto, T., Vanstone, S.: New public key schemes based on elliptic curves over the ring zn. In: LNCS (ed.) Advances in Cryptology - CRYPTO'91, vol. 576, pp. 252–266 (1991)
7. Lillibridge, M., Elnikety, S., Birrel, A., Burrows, M., Isard, M.: a cooperative Internet Backup Scheme. In: Usenix Annual Technical Conference (General Track), pp. 29-41 (2003)
8. Oualha, N., Önen, M., Roudier, Y.: A Security Protocol for Self-Organizing Data Storage. Tech. Rep. EURECOM+2399, Institut Eurecom, France (2008)
9. Oualha, N., Roudier, Y.: A game theoretic model of a protocol for data possession verification. In: IEEE International Workshop on Trust, Security, and Privacy for Ubiquitous Computing (TSPUC'07). Helsinki, Finland (2007)
10. Oualha, N., Roudier, Y.: Securing ad hoc storage through probabilistic cooperation assessment. In: 3rd Workshop on Cryptography for Ad hoc Networks (WCAN'07). Wroclaw, Poland (2007)
11. Schwarz, T.S.J., Miller, E.L.: Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, p. 12. IEEE Computer Society, Washington, DC, USA (2006)
12. Sebe, F., Domingo-Ferrer, J., Martnez-Ballest, A., Deswarte, Y., Quisquater, J.J.: Efficient remote data possession checking in critical information infrastructures. In: IEEE Transactions on Knowledge and Data Engineering. IEEE Computer Society Digital Library (2007)

www.manaraa.com