Email Stamping: Gmelius Blockchain Architecture

Version 1.1 - August 25, 2017

DR. FLORIAN BERSIER
Gmelius, CEO
florian@gmelius.com

RAPHAËL BISCHOF
Gmelius, CTO
raphael@gmelius.com

***Abstract*—We introduce and describe a hybrid architecture which offers a cost-effective and scalable method to prove the existence and integrity of sent emails by anchoring their associated data into the Ethereum blockchain.**

***Keywords*—Email, Blockchain, Cryptography.**

I. INTRODUCTION

Email is more than fifty years old [1]. Originally designed in the mid-1960s, the first message that had the form now recognised as email was sent at the beginning of the 1970s [2]. Since then, email has very little evolved but has progressively gained a predominant place in human exchanges to the point that it is today the most used communication tool in a workplace environment.

While email was originally developed to only send messages, it is now exploited for a larger and more diverse set of purposes: exchange documents, handle sales, manage projects, collaborate with team members, sign contracts, and so on and so forth. This dramatic evolution in terms of usage led to a pronounced mismatch between how email is effectively used and what it is capable of.

A decisive challenge raised by today's email dominance is that the latter means of communication does not offer a built-in function to prove the integrity and authenticity neither of its content nor of its origin. Knowing that emails are increasingly presented as legal evidence in courts worldwide [3], it becomes urgent to think of approaches that could palliate the inherent flaws of email.

This paper introduces and describes a hybrid[1] architecture that rests on the blockchain to 'stamp' emails. More specifically, we propose a cost-effective and secure way to verify the integrity of an email using the Ethereum blockchain [4].

II. PREPARATION OF A TRANSACTION

Upon sending, the entirety of an email (henceforth the 'original message') – i.e., its headers, subject, body and attachments (if any) – is $base64$ encoded and converted into a hash. A typical Gmelius[2] transaction, denoted T_i, regroups a set of hashes, every single hash corresponding to an original message.

[1]We speak of a 'hybrid architecture' since we develop a method where a decentralized, distributed, immutable ledger (i.e., the blockchain) completes a centralized powered system (i.e., the Email).

[2]https://gmelius.com

A. Email Hashing and Signing

The hash of an email is computed using the SHA2-512 hashing algorithm and signed with our own 512-bit private RSA key. The encryption of the hash ensures that the email part of the coming transaction is anchored by Gmelius into a blockchain, and so helps identify the origin of T_i. Put differently,

$$\alpha = \Pi(A), \tag{1}$$

where A is the original message encoded in $base64$, $\Pi(\cdot)$ the hashing and signing function and α the resulting hash.

B. Hashes Aggregation

All those hashes are aggregated into a single hash by building a Merkle tree [5], that is a binary tree in which every hash of an email corresponds to a leaf node, and every non-leaf node results from the hash of the merger of its child nodes.

Merkle trees are commonly used in computing science and cryptography to perform hashes aggregation as they offer an efficient way to verify contents of large data structures. Interestingly, many cryptocurrencies[3] rest on the same model to aggregate transactions into every block of the chain.

Chaining and concatenating hashes in order to insert only the top-level hash into a blockchain offers three advantages:

1) this prevents bottlenecks,
2) this significantly decreases the cost of a transaction,
3) this reduces the amount of data needed to prove an anchorage.

The top-level node in a binary Merkle tree is the hash resulting from the aggregation of all the other hashes in the leafs, and it is called the Merkle root, Ω (the 'root' henceforth). Similarly, the binary chain proving that a specific hash indeed belongs to a tree is called the Merkle proof (the 'proof' henceforth). Finally, the algorithm used to compute the hash in every non-leaf node by combining its two child nodes is called the Merkle mixer, $\pi(\cdot,\cdot)$ (the 'mixer' henceforth).

A mixing function orders the hashes from lowest to highest before the aggregation. To note that when the number of emails (i.e., nodes) to aggregate is odd, the orphan node directly

[3]E.g., Bitcoin, Ethereum, Litecoin

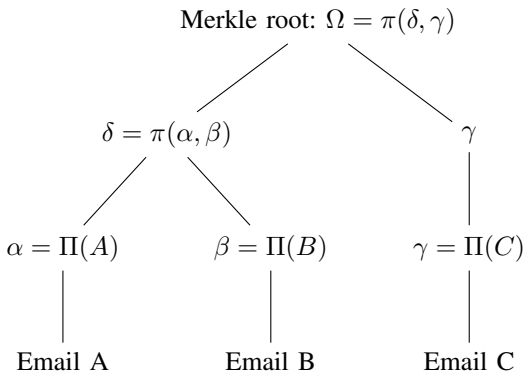

Fig. 1: Example of a Merkle tree with 3 emails and their respective hashes as leaf nodes.

moves to the next level without applying any mixer to its value. The hash resulting from the mixer $\pi(\cdot, \cdot)$ is computed thanks to a standard SHA2-512, which differently from $\Pi(\cdot)$ does not use a private key.

C. Transaction

Finally, the root or top-level hash Ω represents the dataset that will form the transaction T_i to be anchored into a blockchain.

III. Blockchain Anchoring

We decided to use the Ethereum blockchain to anchor our data. This choice rests on three key criteria:

1) A convenient protocol and API;
2) A demonstrable level of maturity;
3) Good performance, i.e., average block time.

TABLE I: Average time betweem blocks. A comparison between the highest capitalization blockchains.

Blockchain	Average block time
Ethereum	$\sim$ 15 seconds
Litecoin	$\sim$ 2.5 minutes
Bitcoin	$\sim$ 10 minutes

A transaction T_i is inserted into the public Ethereum blockchain everytime a Merkle tree contains at most x emails, where x increases with the service use.

On top of Ethereum, the service rests on Parity[4], an Ethereum client that has been chosen for security and performance reasons. The Parity instance runs a Linux server with no RPC/HTTP open ports, and all communications between the Ethereum blockchain and the application is handled using the Web3.js[5] framework, an Ethereum compatible JavaScript API which implements the Generic JSON RPC spec.

A detailed diagram of sequence summarizing the whole architecture can be found in the Appendix.

[4]https://github.com/paritytech/parity

[5]https://github.com/ethereum/web3.js

IV. Proof of Anchorage

Once the transaction has been completed and the root anchored into the Ethereum blockchain, the Gmelius user receives an email that confirms the anchorage and contains a summary of all the data needed to prove that her email belongs to the transaction.

A. Proof formatting

A typical Gmelius proof of anchoring consists of a list containing the following elements:

- The transaction hash that identifies the transaction in the Ethereum blockchain;
- The email in its $base64$ encoded format, e.g., A;
- The email hash resulting from $\Pi(\cdot)$, e.g., α;
- The Merkle root Ω;
- The Merkle proof with all the siblings, e.g., $\{\beta, \gamma\}$;
- the Gmelius public RSA 512-bit key.

B. Proof verification

To verify that an original message is part of a transaction, we start by checking that the email hash corresponds indeed to the original message using the Gmelius public RSA key.

Then, we must reconstruct the path between the email hash and the root. This is done in a sequential fashion, by applying the mixing function to the hash and its first sibling in the proof, then applying the same mixer to the resulting hash and the next sibling, and continuing this process subsequently until there are no more siblings left to concatenate. The proof is complete once the resulting root matches the Merkle root Ω.

Appendix

See Figure 2.

Acknowledgment

The authors would like to thank Jérémy Matos, Software Security Expert at Securing Apps[6], for his insights and recommendations that helped formulate the early architecture presented in this paper.

References

[1] Van Vleck, T., 2001. *The History of Electronic Mail*. http://www.multicians.org/thvv/mail-history.html

[2] Tomlinson, R., 2001. *The First Network Email*. http://openmap.bbn.com/~tomlinso/ray/firstemailframe.html

[3] Mason, S. and Seng, D., 2016. *Electronic evidence*.

[4] Wood, G., 2014. *Ethereum: A secure decentralised generalised transaction ledger*. Ethereum Project Yellow Paper, 151.

[5] Merkle, R. C., 1988. *A Digital Signature Based on a Conventional Encryption Function*. Advances in Cryptology CRYPTO '87. Lecture Notes in Computer Science. 293. p. 369.

[6]https://www.securingapps.com

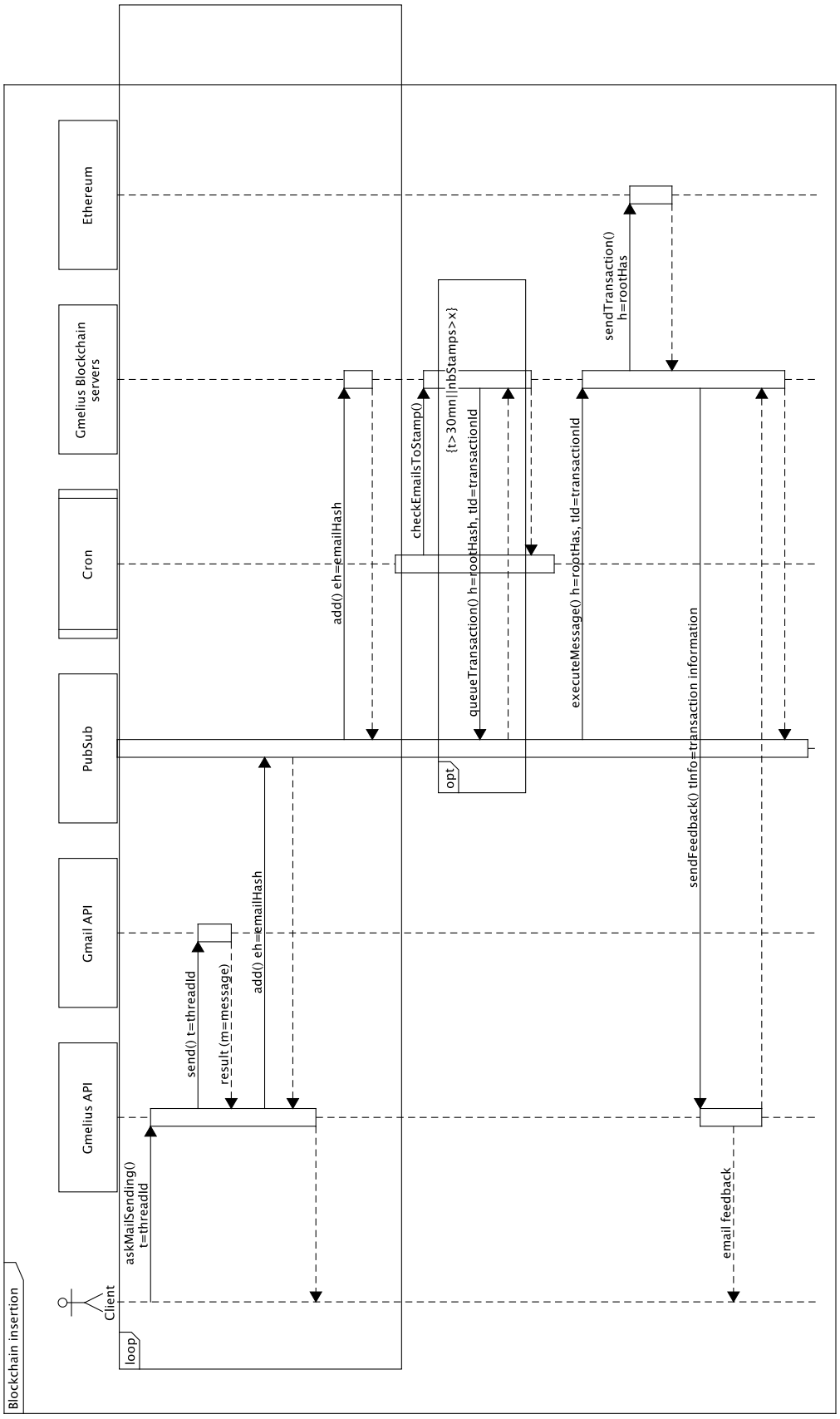

Fig. 2: Diagram of Sequence for the Gmelius Email Stamping Architecture.