Quotable Signatures using Merkle Trees

Michael Kreutzer[1] Ruben Niederhagen[1] Kris Shrishak[2] Hervais Simo Fhom[1]

Abstract: Fake news have been around since time immemorial. But the widespread reach and the rate of propagation through social media websites makes the issue of fake news a grave concern. We propose to address the issue of fake news through the use of quotable signatures using Merkle trees to verify news shared on social media websites.

Keywords: Fake news, Merkle tree, social media.

1 Introduction

Recently, there has been a growing concern about the spread of fake news. Though the phenomenon of fake news is not new, the use of social media websites allows for the fake news to reach a wider audience in a shorter span of time, thus necessitating to suppress them at an early stage. Though there are multiple methods used to spread fake news, in this work, we focus on the following scenario: A journalist at a news organization researches and writes an article and publishes it on the organization's website. Reading this article, a user of a social media website wishes to share it with his/her followers by quoting a part of the text from the news article. Since the origin of the text is from an established organization and since the topic has been well researched by the author, the user wants to include a digital signature of the author or the organization in order to proof the origin of the text. This allows the community to detect well researched contents more easily and to distinct it from fake news. For quoting parts of a text while maintaining its signature, we propose "quotable signatures" using Merkle trees, which is simpler than previously proposed solutions based on machine learning [RSL17; SFR17].

2 Fake News Detection

Our goal is to have a single signature that is valid for the entire text but also for partial quotes of the text. In order to achieve this goal, we propose to use a Merkle tree [Me79]: The input text is split into single words and punctuation marks. Hashes of these tokens are

[1] Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany.
ruben.niederhagen@sit.fraunhofer.de
[2] TU Darmstadt, Darmstadt, Germany.

doi:10.18420/inf2019_64

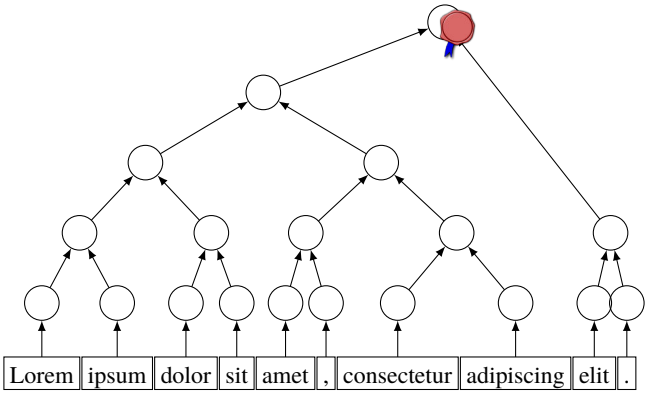

Abb. 1: Merkle signature tree for a short example.

the leaves of the Merkle tree. The Merkle tree over these leaf nodes is a binary hash-tree. The nodes on each level are pairwise-hashes of nodes on the previous level. For computing a quotable signature, the author computes the Merkle tree up to the root node and signs this root with his/her private key. Figure 1 shows an example of a Merkle signature tree for a short example text. The signature of the text can be easily checked by a verifier by re-computing the root of the Merkle tree and by verifying the signature on the root.

When quoting an article, naturally only a (small) part of the original text is provided. Therefore, not all the leaf nodes of the Merkle tree are available and the verifier requires additional information for computing the root node and thus for verifying the signature. However, instead of providing the entire text, it is sufficient to provide the *verification path* in the Merkle tree, i.e., those nodes in the tree that are missing on the path from the quoted text nodes to the root node. Figure 2 shows an example of the verification path if only a few root nodes are provided in the quote. In this example, the verification path consists of the dark-gray nodes. Given the quoted text and these verification nodes, the verifier is able to re-compute the root node and to verify the signature.

The leaf nodes are computed as hashes of the text tokens. The inner nodes are computed as hashes of the concatenated values of their child nodes: The value of the right child node is appended to the value of the left child node. This guarantees that the order of the words in the quote can not be changed by an attacker. Due to the structure of the Merkle tree, the verifier knows which leaf nodes are missing in the quoted text and therefore has information about where the quote is skipping parts of the original text and therefore where context might be missing.

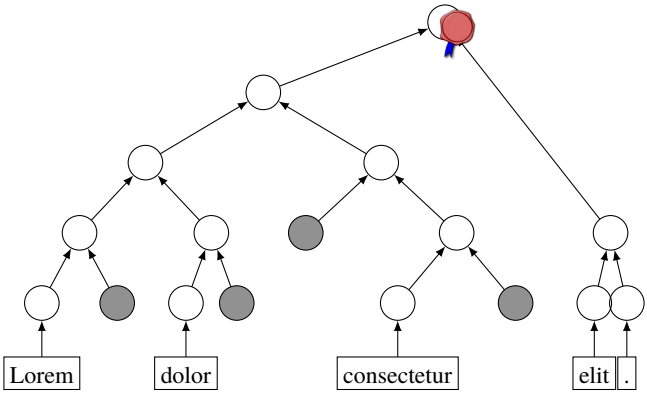

Abb. 2: The gray nodes give the verification path for a set of leaf nodes.

2.1 Cost

Usage of quotable signatures imposes some cost in terms of computation and data size (storage and transmission). These costs occur for signers (authors) that provide quotable signatures for their text, for *quoters* who quote some signed text, and for verifiers (readers) who want to verify quotable signatures on original texts or quotes.

Signers. The computational overhead for signers consists of the computation of the Merkle tree and the generation of the signature on the root node. For a text of n tokens, the height of the tree is $h = \lceil \log(n) \rceil$. There are $2^h - 1$ inner nodes that require concatenated hashes from their child nodes and n leaf nodes that are hashes of the text tokens. Thus, in total, the signer has to compute $2^h + n - 1$ calls to the hash function plus the computation of the signature on the root node.

The increment of the data size for a complete text with quotable signature is moderate. The Merkle tree does not need to be published; only the signature on the root node is needed in addition to the original text. Additional information is required depending on the certification scheme.

Quoters. The cost for the quoters depends on the number of words quoted and the number of sections quoted. For estimating the worst case cost for the quoter, consider the following three cases:

- *One single word* is quoted:
  The entire verification path from the leaf node up to the top needs to be provided.

Therefore, the verifier has to recompute the entire Merkle tree, i.e., $2^h + n - 1$ hashes. The entire verification path of h nodes needs to be provided; the data size grows by h hashes.

- *One single, continuous part* of the text is quoted:
  The exact cost depends on the number of tokens that are quoted. The worst case is a quote of two words in the middle of the Merkle tree, requiring independent verification paths almost up to the root node. In this case, the quoter needs to compute $2^h - 1 - 2\lceil \log_2(n/2) \rceil$ hashes. The number of nodes in the verification path is $2\lceil \log_2(n/2) \rceil$.
- *Several parts* are quoted:
  The number of nodes that need to be re-computed in the Merkle tree shrinks the more tokens are quoted. However, the worst case for the data size of the verification path is when every second word is quoted. This requires $n/2$ nodes in the verification path. (In this case, it is more efficient to provide the original tokens instead of their hashes.)

Therefore, the worst case with regards to computational cost is the re-computation of almost the entire Merkle, i.e., $2^h - 1 - \lceil \log_2(n) \rceil$ hashes. The worst case for the data size is to quote every second word, resulting in $n/2$ hashes.

Verifiers. In the worst case, the verifier needs to re-compute the entire Merkle-tree with n leaf nodes and $2^h - 1$ inner nodes This results in a cost of $2^h + n - 1$ calls to the hash function. The more nodes are provided in the verification path, the less hashes need to be re-computed by the verifier. In any case, the verifier needs to verify the signature on the root node and the certificate of the signer.

2.2 Statistics and Efficiency

In 1951, Shannon claimed that the average word length in English texts is 4.5 characters [Sh51]. Recent studies show that in modern texts the average word length is slightly larger[3]. In the following, we use 4 characters as the lower bound for a worst-case estimation.

For 256-bit hash functions, each 256-bit (32 byte) hash value corresponds to 32 characters in ASCII encoding. However, for embedding quotable signatures into HTML code, the hash values need to be encoded as text. For compatibility, we investigate Base64 encoding. In Base64 encoding, each 256 bit string requires 44 characters which equals $44/4 = 11$ tokens. In other words, each hash value is worth eleven tokens of storage. Therefore, a sub-tree of height up to four may be better represented by words than a hash.

[3] 4.79 characters according to `https://norvig.com/mayzner.html`; 5.1 characters according to `https://www.wolframalpha.com/input/?i=average+word+length`.

Assuming "a single, continuous quote", worst case for verification path length is $2\lceil \log_2(n/2) \rceil$. So, for 24 tokens, the worst case signature equals the length of the text: $24 \cdot 4 = 96 = 2 \cdot \lceil \log_2(12) \rceil \cdot 11 + 2 \cdot 4$. Therefore, the minimum text size for a quotable signature is about 24 tokens or about 96 characters.

2.3 Signatures and Verification

We propose to use a standard signature scheme for signing the root node of the Merkle tree. In order to reduce the overhead of the overall scheme, we recommend to use an elliptic curve (ECC) signature scheme instead of RSA or DSA, for example `ecdsa_secp256r1_sha256` or `ed25519` from the TLS 1.3 draft. This gives an overhead of only 256 bit (32 bytes) for the signature.

For the verification of the root node, the verifier needs to know and trust the public key of the author. This can be achieved by a standard certification scheme.

3 Acknowledgements

The research reported in this paper has been supported in part by the German Research Foundation (DFG) within the project D.3 under RTG 2050 "Privacy and Trust for Mobile Users" and in part by the German Federal Ministry of Education and Research (BMBF) within the project "Scrutinise and Thwart Disinformation (DORIAN)".

Literatur

[Me79] Merkle, R. C.: Secrecy, authentication, and public key systems, Ph.D. thesis, Electrical Engineering, Stanford, 1979.

[RSL17] Ruchansky, N.; Seo, S.; Liu, Y.: CSI: A Hybrid Deep Model for Fake News Detection. In: CIKM. ACM, S. 797–806, 2017.

[SFR17] Singhania, S.; Fernandez, N.; Rao, S.: 3HAN: A Deep Neural Network for Fake News Detection. In: ICONIP (2). Bd. 10635. Lecture Notes in Computer Science, Springer, S. 572–581, 2017.

[Sh51] Shannon, C. E.: Prediction and entropy of printed English. The Bell System Technical Journal 30/1, S. 50–64, Jan. 1951.