Clearing Fuzzy Signatures: a Proof of Work Blockchain Protocol for Biometric Identification

Paolo Santini, Giulia Rafaiani, Massimo Battaglioni, Marco Baldi, and Franco Chiaraluce

Università Politecnica delle Marche, Ancona, Italy
{p.santini, g.rafaiani, m.battaglioni, m.baldi, f.chiaraluce}@univpm.it

In a conventional Proof of Knowledge identification scheme, a user demonstrates to possess some private information (say, a secret key) by replying to the random challenges of a verifier, which involve public data (say, the corresponding public key). A simple paradigm to obtain such a protocol consists in using digital signatures: the private and public data are the signing and verification keys, respectively. Every time a user wants to be identified, he is challenged with a random message, which he must sign. Identification is successful iff the signature is verified by the associated verification key.

In some contexts, signing keys may be *fuzzy*, i.e., associated with a statistical distribution so that distinct samples are, with high probability, slightly different. The most natural example of fuzzy sources is that of biometric data, e.g., fingerprints. Signing with biometric-derived keys would indeed produce signatures that, with very high probability, cannot be verified with the originally enrolled public key.

To reconcile with the public key, one may use techniques such as fuzzy-commitment schemes [1, 2] and fuzzy signatures with linear sketch [3, 4]. However, they all come with some overhead, in the form of additional public data and/or modifications to standard signature algorithms.

Our contribution. We describe a novel Proof of Work (PoW) blockchain protocol which is specifically tailored to identify users holding a fuzzy secret key distribution. We exploit the fact that fuzzy keys produce signatures that are *blurred*, i.e., close to the signature that would be verified by the user's public key. To *clear* the signature, i.e., to remove the noise due to the key fuzziness, the miners go through a brute force search. The key point is in the fact that the cost of this process, which would be too high for a single user, gets distributed among all the miners. In our protocol, transactions are signed with fuzzy keys and verification implies users identification. In other words, the outcome of the consensus protocol is not only the blockchain status update, but also the identification of a user.

The resulting protocol is highly customizable, since features such as average block size and average creation time can be easily customized. With a modification in the clearing process, the protocol is guaranteed to achieve Byzantine Fault Tolerance (BFT); the only drawback is in that each clearing attempts becomes moderately slower. This trade-off is ruled by an integer parameter X: the average block creation time increases as $O(X)$, while the BFT is $\frac{1}{2+1/X}$. In this abstract we give a simple description of the protocol, by considering the Elliptic Curve Digital Signature Algorithm (ECDSA).

A prototype of the proposed scheme

We briefly describe the archetype version of the proposed protocol, based on the ECDSA signature scheme. Let us consider N users $\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_N$. As a structural assumption, we consider that each user $\mathcal{U}_i$ is associated with a distribution (e.g., depending on some user's biometric data) whose mean $\widehat{x}_i$ is uniformly distributed over $\mathbb{Z}_n$ (here, n refers to the additive order of G, the generator of the elliptic curve). To model the fuzziness in a simple way, we assume each distribution is uniform in $[\widehat{x}_i - w; \widehat{x}_i + w]$. In the (initial) enrollment phase:

- a secret key x_i for user $\mathcal{U}_i$ is sampled according to his distribution;
- the public key $\mathsf{pk}_i := Q_i = x_i G$ is computed.

The users' public keys are collected in a public file $\mathscr{L} = \{\mathsf{pk}_1, \mathsf{pk}_2, \cdots, \mathsf{pk}_N\}$. When user $\mathcal{U}_i$ wants to be identified:

1. he sets the raw transaction as m, where m contains all useful data that make the transaction unique (e.g., the current timestamp and the hash of the latest block);
2. he samples a fresh signing key x_i';
3. he computes s' as the signature of m, using x_i'.

The pair (s', m) is then delivered to the network, which starts mining the transaction. It can be easily proved that, if $x'_i = x_i + e$, where e is such that $|e| \leqslant 2w$, then the signature s' would be validated by the public key $Q'_i = x_i G + eG = Q_i + eG$. Notice that Q'_i can be recovered from s'. Then, each miner starts the clearing process, which we summarize as follows:

1. randomly pick a candidate $\widetilde{e}$ for e and compute $\widetilde{Q}_i = Q'_i - \widetilde{e}G$;
2. determine whether $\widetilde{Q}_i \in \mathscr{L}$: if so, produce the transaction as $T = \left(m, s', \widetilde{e}, \widetilde{Q}_i\right)$, otherwise restart from step 1.

Upon the reception of T, each blockchain node checks that i) T is properly constructed by checking that m correctly references the latest block and the timestamp is trustworthy, ii) $|\widetilde{e}| \leqslant 2w$, and iii) $Q'_i - \widetilde{e}G \in \mathscr{L}$. If the above conditions are met, then T is included in the blockchain and the miner gets rewarded.

Security: cryptographic primitives

In the described context, given a public key Q, the problem of forging a signature becomes that of finding a signature s^* that is verified by some Q^* within distance $2w$ from Q. It can be shown that this problem is essentially as hard as forging a (non fuzzy) ECDSA signature[1].

Consensus protocol: Byzantine Fault Tolerance (BFT) and time complexity

As any PoW-based blockchain, the considered paradigm is secure if a sufficiently large portion of miners is honest. In the mining process, candidates for e will be sampled from a cryptographically secure PRNG. This forces miners to go through an exhaustive search, in order to find a valid candidate for e. When the computational cost of the PRNG is not too small, say, it is $X \cdot t_{\mathsf{ECDSA}}$ (i.e., X times the cost of computing an ECDSA signature), the protocol achieves a BFT tolerance of $\frac{1}{2+1/X}$.

The cost of the mining process depends on w: when there are M online miners, the average time to mine a transaction is in

$$O\left(\min\left\{\frac{4w+1}{M}; 1\right\} \cdot \left(X + 1 + \log_2(N)\right)\right).$$

Since every transaction in the block needs to be verified, the average block size can be tuned, according to the values of w, N and M. Also, a traditional PoW step can be added, in case some blocks contain very few transactions and, as such, are too easy to be mined.

References

[1] M. Baldi et al. "On Fuzzy Syndrome Hashing with LDPC Coding". In: *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*. ISABEL '11. Barcelona, Spain: Association for Computing Machinery, 2011.

[2] A. Juels and M. Wattenberg. "A Fuzzy Commitment Scheme". In: *Proceedings of the 6th ACM Conference on Computer and Communications Security*. CCS '99. Kent Ridge Digital Labs, Singapore: Association for Computing Machinery, 1999, pp. 28–36. ISBN: 1581131488.

[3] S. Katsumata et al. "Revisiting Fuzzy Signatures: Towards a More Risk-Free Cryptographic Authentication System based on Biometrics". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 2046–2065.

[4] K. Takahashi et al. "Signature schemes with a fuzzy private key". In: *International Journal of Information Security* 18.5 (2019), pp. 581–617.

[1] Namely, an oracle solving the fuzzy ECDSA forging problem can be used to forge a non-fuzzy signature, using only $O(w)$ queries.