Pre-signed redacted text signature verification via NIZKP

Noah Ostle

February 26, 2026

1 Design

Aim: To be able to take a message M signed by an original signer eg. Alice, redact it to M', publish it, and prove that it was signed by Alice without revealing the redacted portion of the message.

Design Goals: Essentially, we want a *non-interactive zero knowledge proof* that convinces a verifier;

- There exists a message M whose hash was signed by Alice.
- The public part of the message P concatenated with the redacted part of the message w equals M.

Without revealing;

- The redacted text, w
- The hash/signature of M, as doing so would allow an attacker to brute-force guess w by making a guess w', and comparing the hash of $P|w'$ with the hash of M.

Scheme:

Let $H : \{0,1\}^* \to \{0,1\}^{256}$ be a 256-bit cryptographic hash function, eg. SHA-256.

Let $V : PK_a \times \{0,1\}^{256} \times \sigma_a(M) \to \{0,1\}$ be a signature verification algorithm, where PK_a is Alice's public key, and $\sigma_a(M)$ is Alice's signature on message M. Eg. a RSA-PSS or PKCS signature.

First, we let Alice's original message be M, and from it, decide what we want to publish; P, and what we want to redact; w.
For example we may want to prove we solved a puzzle, but not reveal our answer;

> M = Congratulations! Your answer 'Sup3R53cR3t3301!' was correct.
>
> P_1 = Congratulations! Your answer, P_2 = was correct.
>
> w = 'Sup3R53cR3t3301!'

Note: you can continue to split up the text into multiple P_n's and w_i's, but for the purpose of this explanation we will just write P, w without indices st. $M = P||w$ to save space.

If we imagine M, and subsequently P and w as byte strings; $P, w \in \{0,1\}^*$, then we can naturally define the hash of the message h_M as;

$$h_M := H(M) = H(P||w)$$

We can now generate criteria for a non-interactive zero knowledge proof π:

Public inputs: $PK_a,\ P$
Witness: $(w,\ h_M,\ \sigma_a(M))$

Relation: $R((PK_a,\ P),\ (w,\ h_M,\ \sigma_a(M))) = 1 \iff$
$(h_M = H(P||w))\ \wedge\ (V(PK_a,\ h_M,\ \sigma_a(M)) = 1)$

ie. The relation is true when the hash of the public message and the redacted portion concatenated is equal to the hash of M, and there exists a valid signature by Alice for the hash of M.

$\therefore$ A verifier should accept the proof iff:

$\exists(w,\ h_M,\ \sigma_a(M))\ :\ (h_M = H(P||w)) \wedge (V(PK_a,\ h_M,\ \sigma_a(M)) = 1)$

As you can see, the relation R relies on the hashing function H, and the signature verification function V.
In theory, we could choose any hashing algorithm and signature algorithm, but for this example we will use RSA PKCS #1 v 1.5 and SHA256.
If we can prove that these algorithms can be performed on a fixed input in polynomial time, then we can construct a satisfiable boolean circuit for use with ZK-SNARK machinery.
Firstly, we will take the example of a textbook RSA signature. RSA signature verification involves computing $\sigma^e \texttt{ mod } n$ using modular exponentiation as specified in PKCS #1 v2.2 (RFC 8017) [1]. The dominant cost is modular exponentiation on a k-bit modulus. With naive integer arithmetic, modular multiplication is $O(k^2)$ bit operations and square-and-multiply exponentiation uses $O(log\ e)$ such multiplications. Thus, when the public exponent e is fixed or bounded (as in best practice), RSA verification runs in $O(k^2)$; Additionally, SHA-256 runs in $O(n)$ as per NIST FIPS PUB 180-4 [2] **TODO: describe the ZKSNARK circuit design**

References

[1] K. Moriarty, B. Kaliski, J. Jonsson, and A. Ruschmann, "PKCS #1: RSA Cryptography Specifications Version 2.2," Internet Engineering Task Force (IETF), Nov. 2016, rFC 8017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8017

[2] National Institute of Standards and Technology, "Secure Hash Standard (SHS), FIPS PUB 180-4," U.S. Department of Commerce, National Institute of Standards and Technology, Federal Information Processing Standard Publication 180-4, Aug. 2015, available free of charge. SHA-1 and SHA-2 family specs. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf