Hashing into quadratic residues
modulo a safe prime composite

Sietse Ringers

February 25, 2021

Set \( n = pq \), and set \( p', q' \) such that \( p = 2p' + 1 \) and \( q = 2q' + 1 \). Suppose \( p, q \) are safe primes, i.e., \( p' \) and \( q' \) are also prime. Call a function \( H \) a cryptographic hash function if it is (second) pre-image resistant and collision resistant. Given a cryptographic hash function \( H \) whose output is sufficiently long, it is possible to define another hash function \( H_n \) as the composition of \( H \) and squaring modulo \( n \). This document proves in section 1 that then \( H_n \) is also a cryptographic hash function, after first showing three preliminary propositions that we use in our proof. Then in section 2 we provide an explicit description of the hash function \( H_n \) in pseudocode.

We write \( \mathbb{Z}_n^* \) for \((\mathbb{Z}/n\mathbb{Z})^*\), the multiplicative group of the integers modulo \( n \) having an inverse (i.e. \( 0 < x < n \) with \( \gcd(x,n) = 1 \)). Additionally we write \( QR_n = (\mathbb{Z}_n^*)^2 = \{ x^2 \mid x \in \mathbb{Z}_n^* \} \) for the group of quadratic residues modulo \( n \).

1 Proving security

**Proposition 1.** \( \mathbb{Z}_n^* \) contains exactly four square roots of 1, i.e. elements \( X \) such that \( X^2 = 1 \), namely:

- \( 1 \mod n \)
- \( n - 1 \mod n \)
- \( R := Pp - Qq \mod n \), where \( P, Q \) are the integers such that \( Pp + Qq = 1 \), given by the extended Euclidean algorithm
- \( n - R \mod n \)
Proof. It is clear that 1 and \( n - 1 \) are square to 1 mod \( n \). Due to the Chinese Remainder Theorem (CRT), \( \mathbb{Z}_n^* \) is isomorphic to \( \mathbb{Z}_p^* \times \mathbb{Z}_q^* \). The order of the two group factors is \( p - 1 = 2p' \) and \( q - 1 = 2q' \) respectively. Both of those groups have a subgroup of order 2, namely the ones generated by \(-1 \mod p\) and \(-1 \mod q\) respectively, and because of Lagrange’s theorem those must be the only such subgroups. Therefore because of the CRT isomorphism, \( \mathbb{Z}_n^* \) has two distinct subgroups of order two, generated by \((1, -1)\) and \((-1, 1)\). Under the inverse of that isomorphism these are \( R = Pp - Qq \mod n \) and \(-R = n - R \mod n \).

Proposition 2. Any quadratic residue \( Y = X^2 \mod n \in QR_n \) unequal to 1 has exactly four square roots in \( \mathbb{Z}_n^* \), namely \( X \mod n \), \( n - X \mod n \), \( RX \mod n \) and \( n - RX \mod n \). Two of these have representatives smaller than or equal to \((n - 1)/2\).

Proof. That the mentioned numbers square to \( Y \) is easily seen using direct computation. Furthermore, if \( X \) and \( Z \) have the same square mod \( n \), then \( X/Z \mod n \) squares to 1 mod \( n \), so if any \( Y \) had more than four distinct roots then this would yield a fifth square root of one which does not exist.

As to the second claim of the proposition, one of \( X \) and \( n - X \) must be smaller than or equal to \((n - 1)/2\), and then the other will be larger. The same must hold of the smallest representatives of \( RX \mod n \) and \( n - RX \mod n \).

Proposition 3. If one knows one of the nontrivial square roots of 1 \( \in QR_n \) (i.e. not 1 or \( n - 1 \)), then one can factor \( n \).

Proof. Denote the square root again with \( R \). Since \( R^2 = 1 \mod n \) we have \( R^2 - 1 = (R + 1)(R - 1) = 0 \mod n \); i.e. for some integer \( a \), \( (R + 1)(R - 1) = an = apq \), with \( a \neq 0 \) since \( R \neq 1 \). Now since \( p \) is prime, it must divide one of the two factors, \( R + 1 \) or \( R - 1 \). Since \( R + 1 \neq pq = n \) (as we assumed the square root was nontrivial), it follows that \( q \) must divide the other factor. So the factors of \( n \) are \( \gcd(n, R - 1) \) and \( \gcd(n, R + 1) \).

Since \( p = 2p' + 1 \) and \( q = 2q' + 1 \) are safe primes, the order of \( \mathbb{Z}_n^* \) is \( \phi(n) = (p - 1)(q - 1) = 4p'q' \). Then it is easy to see that the order or \( QR_n \) equals \( p'q' \). For example, using CRT, the fact that \( \mathbb{Z}_p^* \) and \( \mathbb{Z}_q^* \) are cyclic, and then CRT again, we have

\[
\mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^* \cong \mathbb{Z}_{2p'} \times \mathbb{Z}_{2q'} \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_{p'} \times \mathbb{Z}_{q'}
\]

Then the quadratic residues in \( \mathbb{Z}_n^* \) are those whose two components in the two group factors \( \mathbb{Z}_2 \) equal 0. This means that the order of \( QR_n \) is indeed \( p'q' \).
Theorem 1. If \( f : \{0,1\}^* \rightarrow QR_n \) is any function such that \( f(x) = 1 \) does not happen or happens with negligible probability, and if factoring is hard, then its output will be a generator of \( QR_n \) with overwhelming probability.

Proof. Suppose \( G = f(x) \) is not a generator; that is, its order is not the maximal order \( p'q' \). Without loss of generality let its order be \( p' \), so that \( 1 = G^{p'} \mod n \). Since \( n = pq \), reducing modulo \( p \) gives the identity

\[
1 = G^{p'} \mod q = (G \mod q)^{p'} \mod q.
\]

Now \( G \mod q \) is an element of \( \mathbb{Z}_q^* \), whose group order is \( 2q' \), and since \( G \) is a quadratic residue the order of \( G \mod q \) cannot be \( 2q' \), so it must be either 1 or \( q' \). In the latter case, our identity \( (G \mod q)^{p'} = 1 \mod q \) would imply that \( q' \) divides \( p' \) which is impossible because \( p' \) is prime. Therefore, the order of \( G \mod q \) in \( \mathbb{Z}_q^* \) is 1, i.e. \( G = 1 \mod q \). This implies that \( G - 1 = aq \) for some \( a \), i.e., \( \gcd(n,G - 1) = q \).

Theorem 2. Let \( H : \{0,1\}^* \rightarrow [2,(n-1)/2] \) be a cryptographic hash function (i.e. it is collision resistant and (second) pre-image resistant). Define \( H_n : \{0,1\}^* \rightarrow QR_n \) by \( H_n(x) = H(x)^2 \mod n \). If factoring is hard, then \( H_n \) is also a cryptographic hash function, which outputs generators of \( QR_n \) with overwhelming probability.

Proof. First note that the output of \( H \) will have with overwhelming probability a multiplicative inverse \( \mod n \), i.e. \( \gcd(H(x), n) = 1 \), because if not, then \( \gcd(H(x), n) \) will factor \( n \). So with some abuse of notation, we may consider the range of \( H \) to be a subset of \( \mathbb{Z}_n^* \), so that the range of \( H_n \) is indeed \( QR_n \).

Suppose \( H_n \) is not collision resistant, so let \( x_1 \neq x_2 \) be such that \( H_n(x_1) = H_n(x_2) \mod n \). Then by Proposition 2, \( H(x_1) \) equals \( H(x_2) \) or \( n - H(x_2) \) or \( RH(x_2) \mod n \) or \( n - RH(x_2) \mod n \). It cannot be \( n - H(x_2) \) since that exceeds \( (n-1)/2 \). Similarly, of \( RH(x_2) \mod n \) and \( n - RH(x_2) \mod n \), only one will have a smallest representative that is smaller than \( (n-1)/2 \). Suppose without loss of generality that it is \( RH(x_2) \). Summarizing, then, we have either \( H(x_1) = H(x_2) \) or \( H(x_1) = RH(x_2) \). Now in the latter case we have \( H(x_1)/H(x_2) = R \mod n \): one of the nontrivial square roots of 1 (since \( 1 < H(\cdot) < n - 1 \)). So if the latter case holds with non-negligible probability, then we have a non-negligible chance of being able to factor \( n \), by Proposition 3. Thus we must have \( H(x_1) = H(x_2) \). So any algorithm that breaks collision resistance of \( H_n \) can be used to break that of \( H \), which is impossible since we assumed \( H \) to be collision resistance.

Collision resistance implies second pre-image resistance. Ordinary pre-image resistance can be proven with an almost identical argument as above.
The fact that \( H_n \) outputs generators with overwhelming probability is proven in the previous theorem.

Because the hash function is the composition of \( H \) and squaring modulo \( n \), and because for each integer smaller than or equal to \((n - 1)/2\) there is exactly one other such integer that squares to the same quadratic residue by Proposition 2, \( H_n \) has exactly twice as much collisions as \( H \) itself. This is to be expected, however, since the range of \( H_n \) is half as large as the maximal range of \( H \) (which we take to be the lower half of \( \mathbb{Z}_n^* \), as above). Additionally, the fact that all quadratic residues have exactly two roots smaller than the upper bound ensures that this non-injectiveness of the square function does not cause particular values of \( QR_n \) to be returned by \( H_n \) more often than others. Summarizing, the output of \( H_n \) “appears as random” as can be expected.

## 2 Instantiation and implementation

In this section we describe the cryptographic hash function \( H_n \) in more detail. For convenience, we interpret the output of our hash functions as large integers; that is, we assume an implicit conversion of the output bytes to integers.

Generally the construction below can be done for any hash function \( H \) of sufficiently long output length, but for concreteness we take \( H(x) = \text{SHAKE256}(x, d) \). Here \( \text{SHAKE256} \) from the SHA3 function family is a so-called Extendible Output Function (XOF): a function that has variable output length, specified in bits as the second parameter \( d \), with the property that for any fixed \( d \) the function \( \text{SHAKE256}(\cdot, d) \) is a cryptographic hash function, and moreover if \( d' > d \) then the first \( d \) bits of \( \text{SHAKE256}(\cdot, d') \) coincide with \( \text{SHAKE256}(\cdot, d) \).

Let \( L_n = |n| \) be the length in bits of the modulus (i.e. 1024, 2048 or 4096). As the theorem above states, for the security of \( H_n \) it is important that the cryptographic hash function \( H \) has the appropriate maximum output; specifically, its output should be smaller than or equal to \((n - 1)/2\). Now since \( \text{SHAKE256}(\cdot, d) \) outputs \( d \) bits the upper limit of its output is \( 2^d \) instead of \((n - 1)/2\). Setting \( d = L_n - 1 = |(n - 1)/2| \), our hash functions will thus sometimes output an integer smaller than \( 2^d \) but larger than \((n - 1)/2\). We can “fix” that by prepending our input bytes with a counter \( i \) starting at 0, i.e. when hashing \( x \) we return \( H(0||x) \) if that is below the upper bound, and if it exceeds \((n - 1)/2 \) we increment \( i \) until \( H(i||x) \leq (n - 1)/2 \). We do the same in the (unlikely) case that \( H \) outputs 0 or 1. To prevent attacks
where $x$ is crafted with a specific $i$ as its first few bits, one should use an encoding such as DER-ASN1 for $i||x$. Note that this does not mean that an implementation has to include a generic ASN1 parser; instead one can work out once and then hardcode the bytes of a DER encoding of the following ASN1 sequence:

$$\text{HashInput ::= SEQUENCE \{ }$$
$$i \text{ INTEGER, }$$
$$x \text{ OCTET STRING }\}$$

Finally, in implementations it might only be possible to specify the output length of SHAKE256 in bytes instead of in bits. In this case, one can simply request $L_n/8$ output bytes and then discard the rightmost bit to end up with the required $d = L_n - 1$ bits.

A description of the algorithm computing $H_n$ summarizing the above may be found in pseudocode below. We assume there that SHAKE256 takes its output length as the second parameter in bits.

---

**Algorithm 1** Cryptographic hash function $H_n : \{0, 1\}^* \rightarrow QR_n$

function $H_n(x)$

$i \leftarrow 0$

repeat

\[ O \leftarrow \text{SHAKE256}(\text{DER-ASN1}(i, x), L_n - 1) \]

\[ i \leftarrow i + 1 \]

until $1 < O \leq (n - 1)/2$

return $O^2 \mod n$

end function

---