Digital Signature Using DLP

Objective

Build a digital signature scheme using the Discrete Logarithm Problem(DLP) and hash functions. Design the collision-resistant hash functions also using DLP.

Zero-Knowledge Proof for DLP

The DLP based digital signature scheme can be designed using the zero knowledge proof for DLP.

Given prime p, generator g, public key $y = g^x \ mod \ p$, the prover(P) needs to prove the knowledge of the private key x to the verifier(V) without revealing it. This can be done by running multiple rounds of the following algorithm:

1. P chooses a random number r, $0 \le r < p-1$ and sends $h = g^r \ mod \ p$ to V.
2. V sends back a randomly chosen bit b.
3. P sends $s = (r + b * x)(mod(p-1))$ to V.
4. V computes $g^s \ mod \ p$, which should be equal to $(h * y^b) \ mod \ p$.

Correctness

- If $b = 0$, P sends $s = r$ to V, and V checks whether $g^s \ mod \ p = h$, i.e, P knows the discrete log of h, which is r.
- If $b = 1$, P sends $s = (r + x)(mod(p-1))$ to V, and V multiplies h, which is $g^r \ mod \ p$ and y, which is $g^x \ mod \ p$ and compares the product $(h * y)(mod \ p)$ with $g^s \ mod \ p$, which is $g^{(r+x)(mod(p-1))} \ mod \ p$.

Hardness

P never sends x to V, only r or $(r + x)(mop(p-1))$, both of which look random. When $b = 1$, V knows that $s = (discreteLog(h) + x)(mod(p-1))$, but it does not know $discreteLog(h)$, which is sent only when $b = 0$, and thus cannot obtain x. Thus, depending upon the choice of the random bit b, V gets either s or $r = discreteLog(h)$, but never both, and cannot compute x.

Now, consider an adversary A, who knows g, p, y but does not know x, trying to convince V that he knows x.

- If $b = 0$, then A just needs to send $s = r$, where r was generated by A in step 1, to convince V, as V will be able to check $h = g^s \ mod \ p$. But if $b = 1$, A is stuck because he does not know x, and cannot generate an s in polynomial time that can satisfy $g^s \ mod \ p = h * y \ (mod \ p)$, because that would mean finding the discrete log of $(h * y)$, which is known to be hard.

- A can cheat V when $b = 1$ by sending $H = (g^r * y^{-1}) \ mod \ p$ to V instead of $(g^r \ mod \ p)$. If V sends $b = 1$, A can send the random number r as s and it will satisfy $g^s \ mod \ p = H * y \ (mod \ p)$. However, if V sends $b = 0$, A cannot fool V, since it doesn't know the discrete log of H, which is supposed to be sent when $b = 0$.

In either case, there is a 50% probability of fooling the verifier V. With a large number of rounds, say k, the probability of successfully cheating in all rounds becomes 2^{-k}. So, for large k, if V could verify correctly in all rounds, there is a very high probability that the prover is not an adversary, and indeed knows x.

Digital Signature Scheme

The proof described above can be used to produce digital signatures, while assuming only the hardness of the discrete log problem.
In the protocol above, the prover P first picks a random r and then the verifier V picks a random b. It's important that P picks first and sends h to V. Otherwise, if P saw V's choice of b, he could cheat in one of the two ways given earlier. For the signature protocol, the signer simulates both the prover and verifier.
The signer first does a random choice of r for P as before, but V's random choice is simulated by hashing the message to be signed and a value computed from P's choice of r.
However, since the signer contains both the prover and the verifier, if b is a single bit, the signer could just generate a few $r's$ until he finds one that produces $b = 0$ from the hash function at V's side, so that authenticity can be claimed without the knowledge of x.
Note that in the zero-knowledge proof, there was a 50% chance of the adversary proving the knowledge of x. To overcome this, we had multiple rounds of the procedure. So, we modify the signing protocol such that instead of a single bit, V now chooses a large integer c.
The protocol is as follows:

1. Assume that the full message M has already been shortened to an digest value m

2. Let x be the secret key known only to the signer. Let p be a large prime, and g be the generator of $\mathbb{Z}_p*$. $(g, p, g^x\ (mod\ p))$ can be published as the public key.

3. In order to sign m, the signer chooses a random r, then computes c using a hash function (simulating V's choice).

$$c = H(m^x\ mod\ p,\ m^r\ mod\ p,\ g^r\ mod\ p)$$

where H is the collision-resistant hash function.

4. Let $s = c * x + r$. Signer publishes the digital signature which is m along with $(s,\ m^x\ mod\ p,\ m^r\ mod\ p,\ g^r\ mod\ p)$

5. To check the signature, a verifier first computes c as the hash of the values $m^x\ mod\ p,\ m^r\ mod\ p,\ g^r\ mod\ p$, which were published in the signature. Then the verifier checks

$$g^s\ mod\ p = (g^x)^c * (g^r)\ (mod\ p)$$

and

$$m^s\ mod\ p = (m^x)^c * (m^r)\ (mod\ p)$$

Correctness

The goal is to convince the verifier that you know what x is. s depends on x but doesn't allow the verifier to obtain x because it is multiplied by a random value c and added to a random value r. You can safely tell the verifier $g^x\ mod\ p$ and $g^r\ mod\ p$, because discrete log is hard, and so x and r cannot be obtained from these values. Since the hash function is collision resistant, another set of inputs giving the same value c cannot be found by a PPTM.
Knowledge of the hashed value c is what proves the signer's authenticity. If the signer didn't know x, which is used to obtain c, then trying to find an s satisfying $g^s\ mod\ p = (g^x)^c * (g^r)\ (mod\ p)$ is a general instance of the discrete log problem which is hard. The signer cannot cheat by generating many r values, because there are too many possible c's, and the odds of a c satisfying the tests by chance are negligible.
The checks done by the verifier establish that the signer knows x.

Collision Resistant Hash Function

Consider hash function $h : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ with DLP parameters: generator g of Z_p^*.

$$h(x, y) = g^x * z^y (mod\ p)$$

where z is some element of Z_p^*. Let

$$z = g^k (mod\ p)$$

We will prove that if an adversary can find a collision in this hash function, then that adversary can solve DLP in polynomial time. Since we assume that DLP is hard, no such PPTM adversary exists.

Let $x_1, y_1, x_2, y_2 \in \{0,1\}^n$, such that $x_1||y_1 \neq x_2||y_2$ and $h(x_1, y_1) = h(x_2, y_2)$

$$\implies g^{x_1} * z^{y_1} (mod\ p) = g^{x_2} * z^{y_2} (mod\ p)$$

$$\implies g^{x_1\ -\ x_2} = z^{y_2\ -\ y_1}$$

$$\implies g^{x_1\ -\ x_2} = g^{k(y_2\ -\ y_1)}$$

$$\implies (g^{x_1\ -\ x_2})^{(y_2\ -\ y_1)^{-1}} = (g^{k(y_2\ -\ y_1)})^{(y_2\ -\ y_1)^{-1}}$$

$$\implies k = \frac{(x_1\ -\ x_2)}{(y_2\ -\ y_1)}\ (mod\ (p-1))$$

Thus, we have found the discrete log of z
Since we have assumed DLP to be hard, the assumption of collision was wrong.

$\implies h$ is collision resistant.

Further, we can obtain an arbitrary length hash function using MDT.