Let n, p, G be the parameters of secp256k1. Let d_A be the private key $\in [1, n-1]$, z be the hash of the message, (r, s) the signature corresponding to the private key d_A where $r, s \in [0, n-1]$, and k the corresponding nonce.

Then it holds that $s = k^{-1}(z + rd_A) \mod n$. The obscuren/secp256k1-go package chooses k to be $z \oplus d_A$ (xor). At first glance this seems to be ok, since k is unique for each message and it is unpredictable. An attacker can not directly influence z because it is the outcome of a hash function. However, if an attacker obtains multiple signatures, the reuse of d_A becomes a problem because k becomes predictable. Note that $a \oplus b = a + b - 2(a \wedge b)$.

$$
\begin{aligned}
s &= k^{-1}(z + rd_A) \\
&= (d_A \oplus z)^{-1}(z + rd_A) \\
&= (d_A + z - 2(d_A \wedge b))(z + rd_A) \mod n \\
\iff d_A s + zs - 2s(d_A \wedge z) &= z + rd_A \mod n \\
\iff (s - r)d_A &= (1 - s)z + 2s(d_A \wedge z) \mod n \\
\iff d_A &= ((1 - s)z + 2s(d_A \wedge z))(s - r)^{-1} \mod n
\end{aligned}
$$

Assume that the attacker obtains a second signature (r', s') over z'. Then it holds that

$$
\begin{aligned}
d_A - d_A &= ((1 - s)z + 2s(d_A \wedge z))(s - r)^{-1} - ((1 - s')z' + 2s'(d_A \wedge z'))(s' - r')^{-1} \\
\iff 0 &= (1 - s)z(s - r)^{-1} + 2s(d_A \wedge z)(s - r)^{-1} - (1 - s')z'(s' - r')^{-1} - 2s'(d_A \wedge z')(s' - r')^{-1}
\end{aligned}
$$

Using the fact that $\kappa(d_A \wedge z) - \kappa'(d_A \wedge z') = (\kappa z - \kappa' z') \wedge d_A$ it holds that

$$
\begin{aligned}
0 &= (1 - s)z(s - r)^{-1} - (1 - s')z'(s' - r')^{-1} + (2(s(s - r)^{-1}z - s'(s' - r')^{-1}) \wedge d_A) \\
(1 - s')z'(s' - r')^{-1} + (1 - s)z(s - r)^{-1} &= 2(sz(s - r)^{-1} - s'z'(s' - r')^{-1}) \wedge d_A \\
\alpha &= \beta \wedge d_A
\end{aligned}
$$

That means with two signatures we learn some bits of d_A, namely those positions where β is 1. Assuming that β is distributed uniformly random, the probability that we recovered the bit b after σ signatures is d_A is $(1 - (\frac{1}{2}^\sigma))$. The probability for recovering the whole key is $p(\sigma) = (1 - (\frac{1}{2}^\sigma))^2 56$. $p(5) = 0.0002, p(10) = 0.78, p(15) = 0.99$. Even two signatures can reduce the keyspace substantially.