Cryptography is the science of communicating in the presence of a powerful adversary who is capable of manipulating messages as they travel from sender to receiver. One of the key goals of cryptography is to allow communicating parties to verify that a received message is from the supposed sender and is not altered after being sent. This goal is called *message authentication*. To enable message authentication, we assume that the sender and receiver share some secret, such as a random binary string or some choice of a particular function. The adversary does not know the secret, but can be assumed to know everything else about the authentication scheme. Therefore, it is the secret which must be leveraged to authenticate messages.

Let's say that Alice wants to send Bob a message m, and that Alice and Bob share a secret k. Authentication is most often achieved by Alice generating an authentication tag t as a function of m and k — that is $t = \mathsf{TagGen}(k, m)$ — and Alice sending (m, t) to Bob. What Bob receives is (m', t') where (m', t') will not equal (m, t) if the adversary altered anything in transmission. Bob computes $t'' = \mathsf{TagGen}(k, m')$, and accepts m' as genuine if and only if $t' = t''$. This scheme works only if it is difficult for an adversary to produce a legitimate (m, t) pair without knowing secret k.

To measure the security of the authentication scheme, we imagine that an adversary is given a black-box tag-generator that has been initialized with a random secret k. The adversary is allowed to compute q tags for q messages, so the adversary knows legitimate pairs $(m_1, t_1), (m_2, t_2), \ldots, (m_q, t_q)$. (This models an adversary's ability to learn something about the tag generator by either watching message traffic or by tricking a legitimate party to authenticate messages on the adversary's behalf.) We say that the adversary is successful and *forges* a message if it is able to produce a new (m, t) pair where $t = \mathsf{TagGen}(k, m)$. If we can show that the probability of forgery is low — less than $1/2^{64}$ for example — then the authentication scheme is a good one.

Wegman-Carter Authentication

A good authentication scheme is attributed to Mark Wegman and Larry Carter (1981). Let n be a positive integer and $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$. Let $f : \mathbb{Z}^+ \to \mathbb{Z}_n$ and $h : \{0,1\}^* \to \mathbb{Z}_n$ be functions. Then

$$\mathsf{TagGen}((h, f), m_i) = (h(m_i) + f(i)) \bmod n$$

is a good authentication scheme if f is random and h is randomly chosen from an *ε-delta-universal* hash family. In this formulation, (h, f) is the secret shared by sender and receiver, and i is a message sequence number which must be different for each message authenticated. We will give some justification for this claim later, but first let us examine random functions and universal hash families. By saying that f is random, it is meant that f produces a random output for each different input. So, $f(1), f(2), f(3)$ are each equally likely to be any value in $\mathbb{Z}_n$.

A hash family H is a set of hash functions $\{h_1, h_2, \ldots, h_k\}$ all with the same domain and co-domain (in the case of our authentication scheme each h_i has domain $\{0,1\}^*$ and codomain $\mathbb{Z}_n$). Hash family H is ε-delta-universal if for every $a \neq b$ in $\{0,1\}^*$ and c in $\mathbb{Z}_n$, the number of functions h in H with the property $(h(a) + c) \bmod n = h(b)$ is no more than $\varepsilon \cdot |H|$. Put another way, if you first pick a, b, c and then choose h at random from H, there is no more than ε probability that you choose an h where the outputs of $h(a)$ and $h(b)$ differ by c. This probability arises because there are no more than $\varepsilon \cdot |H|$ hash functions in H with this property and so the probability of choosing one of them is no more than $(\varepsilon \cdot |H|)/|H| = \varepsilon$.

Here is an example. Let $n = 3$ and $H = \{h_1, h_2, h_3\}$ where each h_i maps elements of $\{0,1\}^2$ to elements of $\mathbb{Z}_3$, with $h_1 = \{(00,1), (01,0), (10,0), (11,0)\}$, $h_2 = \{(00,1), (01,2), (10,2), (11,1)\}$ and $h_3 = \{(00,2), (01,0), (10,1), (11,2)\}$. We can compute ε by going over every $a \neq b$ and c combination and asking how many of $\{h_1, h_2, h_3\}$ cause $(h(a) + c) \bmod 3 = h(b)$. In this example, when $(a, b, c) = (00, 01, 0)$, there are zero functions that cause $(h(a) + c) \bmod 3 = h(b)$, but when $(a, b, c) = (00, 01, 1)$ there are two: $(h_2(00) + 1) \bmod 3 = h_2(01)$ and $(h_3(00) + 1) \bmod 3 = h_3(01)$, but $(h_1(00) + 1) \bmod 3 \neq h_1(01)$. Enumerating over all $4 \times 3 \times 3 = 36$ allowable combinations of (a, b, c) and

counting, for each triple, how many of $\{h_1, h_2, h_3\}$ make the property $(h(a) + c) \bmod 3 = h(b)$ true, we find ten combinations for which none of the hash function make the property true, 16 combinations where one hash function makes it true, ten combinations where two hash functions makes it true, and zero combinations where all three hash functions makes the condition true. From this, we can make the claim that for all $a \neq b$ in $\{0,1\}^2$ and c in $\mathbb{Z}_3$, there is never more than $2/3$ of all functions that cause $(h(a) + c) \bmod 3 = h(b)$. By definition, this means H is $(2/3)$-delta-universal.

This previous example is not realistic for message authentication. This next scheme is and is in use today. Let $n = 2^{127} - 1$ (a prime number). Let's say that messages to be hashed are first broken into 96-bit chunks, and each chunk is interpreted as an unsigned integer. So, if message m is to be hashed, it is first transformed into $m = (a_1, a_2, \ldots, a_\ell)$ where a_i is an unsigned integer representable using 96-bits (and so always in $\mathbb{Z}_n$). We define

$$h_k(m) = (a_1 k^\ell + a_2 k^{\ell-1} + \cdots + a_\ell k^1) \bmod n \,,$$

and the hash family $\mathsf{Poly127} = \{h_k \mid k \in \mathbb{Z}_n\}$.

To determine ε for this function, assume $m = (a_1, a_2, \ldots, a_\ell)$ and $m' = (b_1, b_2, \ldots, b_\ell)$ are distinct messages and c is in $\mathbb{Z}_n$. We want to gauge how many functions h_k in Poly127 cause $(h_k(m) + c) \bmod n = h_k(m')$. But,

$$\begin{aligned}(h_k(m) + c) \bmod n = h_k(m') &= (h_k(m) - h_k(m') + c) \bmod n = 0 \\ &= ((a_1 - b_1)k^\ell + (a_2 - b_2)k^{\ell-1} + \cdots + (a_\ell - b_\ell)k^1) \bmod n = 0 \,.\end{aligned}$$

This last formulation has at most ℓ solutions (ie, at most ℓ values for k which cause the polynomial to evaluate to zero), which means there are at most ℓ functions in Poly127 causing $(h_k(m) + c) \bmod n = h_k(m')$ for any $m \neq m'$ and c. Since Poly127 has $2^{127} - 1$ functions in it, Poly127 is $(\frac{\ell}{2^{127}-1})$-delta-universal when hashing messages upto 96ℓ bits in length. Choosing a random h_k from Poly127 is done simply by choosing a random k from $\mathbb{Z}_n$.

Why it works. Again let $n = 2^{127} - 1$. Imagine that an adversary knows legitimate pairs $(m_1, t_1), (m_2, t_2), \ldots, (m_q, t_q)$ where $t_i = (h(m_i) + f(i)) \bmod n$ for random function $f : \mathbb{Z}^+ \to \mathbb{Z}_n$ and h randomly chosen from Poly127. The adversary thinks for some time and guesses (m_j, t). What is the probability that $t = (h(m_j) + f(j)) \bmod n$? There are two possibilities to consider: j is an already used message sequence number (ie, $1 \leq j \leq q$), or it is not. If j is a new sequence number, then $f(j)$ is a random number never seen before, and so the chance that $(h(m_j) + f(j)) \bmod n = t$ is exactly $1/n$ no matter what $h(m_j)$ is. On the other hand, if j is a reused sequence number, say $j = i$ for some $0 \leq i \leq q$, then we need to look at the tag that was generated when j was used the first time: $t_i = (h(m_i) + f(i)) \bmod n$, or more usefully $f(i) = (t_i - h(m_i)) \bmod n$ which is also equal to $f(j)$ because $i = j$. We want to gauge the probability $\Pr[t = (h(m_j) + f(j)) \bmod n]$, so substituting for $f(j)$ we get

$$\begin{aligned}\Pr[t = (h(m_j) + f(j)) \bmod n] &= \Pr[t = (h(m_j) + t_i - h(m_i)) \bmod n] \\ &= \Pr[(t - t_i) \bmod n = (h(m_j) - h(m_i)) \bmod n] \\ &\leq \varepsilon \,.\end{aligned}$$

This last step is because $m_i \neq m_j$ and h is from an ε-delta-universal hash family. So, the maximum chance an adversary has in producing a legitimate new (m, t) pair is the maximum of $1/n$ and ε.

Realizing TagGen

We have seen that sender and receiver can authenticate messages by sharing a secret random function f and a secret hash function h from an ε-delta-universal hash family. We have just seen an appropriate hash function. Sender and receiver need only share a random k from $\mathbb{Z}_{2^{127}-1}$ and use h_k from Poly127 for hashing. For the random function, we use a block cipher, which behaves closely enough to a random function to be appropriate. Block ciphers typically require the selection of a 128-bit random string to work, so this string becomes another secret shared by sender and receiver. In the end, sender and receiver need only share 255 secret bits to authenticate their messages.