Understanding Pre-Shared keys and Resumption tickets

In TLS1.3 these two functions are combined. A new session resumption ticket looks like this

```
struct {
     uint32 ticket_lifetime;
     uint32 ticket_age_add;
     opaque ticket_nonce<0..255>;
     opaque ticket<1..2^16-1>;
     Extension extensions<0..2^16-2>;
} NewSessionTicket;
```

This is how a new session resumption ticket is constructed by the server, and sent to the client. The ticket component is encrypted, probably using some AEAD mode of some block cipher, and it contains within it the pre-shared key PSK.

The encrypted ticket component looks like

```
struct {
      opaque iv[12];
      opaque encrypted_state<0..2^16-1>;
      opaque tag[16];
} ticket;
```

Recall that ticket structure is something we can define for ourselves, and we also choose the encryption method (we chose AES_GCM)

The state component, when decrypted, internally looks something like

```
struct {
          uint32 birth;
          uint32 age_add;
          CipherSuite cipher_suite;
          KeyExchangeGroup group;
          opaque psk[];
          ClientIdentity client_identity;
} decrypted_state;
```

where the client identity is only included if client-side authentication happened on the full handshake. Again what we do here is largely our choice. The age_add field is randomly generated by the server.

The nonce is randomly generated by the server and combined with the resumption master key RMS (which emerges at the end of a full handshake) to determine the PSK to be included in the ticket. The ticket including the nonce is sent to the client. The client stores the PSK with ticket. Both client and server calculate PSK=f(RMS,nonce). The PSK in turn is used to generate the “early secret” used to encrypt early data.

The client returns to the server (in the preshared key extension)

```
struct {
     opaque identity<1..2^16-1>;
     uint32 obfuscated_ticket_age;
} PskIdentity;
```

Now the identity here could just be the encrypted ticket component from above. The server can then decrypt it, determine if ticket is still valid, and finally extract the PSK.

Using a Pre-shared key (not arising from resumption)

In this case the client still sends a pre-shared key extension just like before, but now its just

```
struct {
     opaque identity<1..2^16-1>;
     uint32 obfuscated_ticket_age;
} PskIdentity;
```

as before. But now the identity might just be any string, for example a simple two-byte index into a table of keys. Its the identity that the server expects to receive to enable it to retrieve the associated PSK.

But how is server to know that the identity it receives refers to a ticket, or to a PSK?

How do I indicate that identity is an IBE encrypted PSK? Maybe use a zero age obfuscator? Or maybe prepend a single byte to the identity/ticket to indicate if its (a) a ticket, or (b) an IBE PSK, or (c) a database index.

For now, will use a zero obfuscated ticket age.

Or just use a #define to set up the server to expect an IBE PSK? Except I want it to accept tickets as well!