On Crypto API Safety in the Hands of Unskilled Developers

1. What is the problem?

Conceptually, there are two classes of CryptoOperation: "Plain to ciphertext" operations that convert plaintext to data with cryptographic structure, and "Cipher to plaintext" operations that do the reverse.

P2C	C2P
sign	verify
encrypt	decrypt
digest	

The difference is this: P2C operations can meaningfully be done with many different choices of parameters. C2P operations can only be done with a specific set of parameters.

Both of these create problems for developers.

- For P2C operations, the developer must choose how to set multiple parameters, choices that are likely not obvious to someone not skilled in the art.
- For C2P operations, the developer needs to make sure that they keep all the relevant parameters together with protected information.

So we have two problems:
P2C: How to help developers make good choices
C2P: How to help developers keep ciphertext associated to parameters

2. What would a solution look like?

On the face of it, the P2C problem -- choosing parameters -- seems easy to solve. If there are multiple valid sets of parameters, just have the browser / API implementation make the choice on behalf of the developer.

However, this exacerbates the C2P problem, because there are now many ways for the ciphertext to be separated from its parameters. If a web app does not store the parameters with which the ciphertext was computed (relying on the browser's defaults), then if the browser changes defaults, then the app will be unable to decrypt the ciphertext (or validate the signature). Even if the app stores the parameters, then it needs to make sure that the ciphertext is always associated with the correct parameters; the app cannot, for example, send the

ciphertext for storage on a server, but not the parameters.

So in order to solve the P2C problem, we also need to solve the C2P problem. Namely, we need to make it easy by default for apps to keep parameters and ciphertext together. In API terms, that would seem to indicate that the results of a crypto operation should be provided as an object that contains all the relevant parameters (as indeed, CryptoOperation already does). In addition, it would be helpful if this object had a default serialization, to address the issue of parameters getting lost when the object is stored or sent someplace else.

This gives us two solutions to match the two problems:
P2C: Provide browser-chosen defaults
C2P: Provide results in an object with parameters and a serialization

These don't prevent developers from running into problems -- choosing bad IVs, or deleting default parameters from the object -- but it encourages a default life-cycle that should be problem free:
* Process plaintext, get ciphertext+parameters
* Store ciphertext+parameters
* Process ciphertext+parameters, get plaintext

These solutions also don’t get in the way of more advanced developers. You can still specify all the parameters, and still use whatever parts of the object you want.