HTML5's <keygen> versus SKS/KeyGen2

Feature	HTML5 <keygen>	SKS/ KeyGen2	Comment
Support for SW tokens	0.5	☑	<keygen>: A quirky user-interface and essentially zero issuer-control of key parameters, have to date forced most serious PKI-deployers creating their own provisioning software also for soft tokens.
Support for smart cards	0.1	☐ and ☑	<keygen>: Depends on third-party software for pre-personalization and initialization. Device drivers for different platforms tend to be quite different due to the diversity of cryptographic APIs as well as among smart cards. <keygen> have due to this so far only been used by experts in small deployments and probably never for consumers. KeyGen2: Based on the experience above, *the conclusion is that it is infeasible creating a cost-efficient and reliable ecosystem for consumers based on existing smart cards and protocols*. The KeyGen2-mandated container SKS (Secure Key Store), has a fixed API and does not require any pre-personalization or third-party software.
Credential life-cycle management	☐	☑	Dealing with certificate expiration is crucial for large-scale adoption of PKI. KeyGen2/SKS supports multiple issuers *securely sharing a single container* through a VSD (Virtual Security Domain) concept which hides and protects issuers from each other.
PIN-code support	☐	☑	<keygen>: it is up to the *user* optionally defining password protection of keys. Few banks would ever consider such a system. KeyGen2: PINs can be defined by the issuer or be defined as a policy where the user have to create a compliant PIN during the provisioning process. PINs can be per key or cover a group of keys. In addition, PINs may in turn be manageable via PUKs.
OTP support	☐	☑	OTP (One Time Passwords) are quite useful, particularly in mobile phones for secure web-access from public or locked-down computers.
Information Card support	☐	☑	Information Cards represent a technology that can complement PKI and OTP in various federation scenarios where direct authentication is either impractical or undesirable.
Key attribute support	☐	☑	Key attributes have many uses; from VPN profiles to protection data associated with a key.
Logotype support	☐	☑	In just about all other contexts objects are shown to users in the form of icons. Presenting a key as `CN=Luke Skywalker, O=Jedi, C=US` doesn't appear to be particularly user-friendly. Logotypes also support credential branding.

Transaction-based operation	☐	☑	The classical update of a physical credit-card is that you take out the old card from the wallet, cut it in two pieces and insert the new card. SKS performs this operation on virtual credentials as atomic operations leaving you either with only a new “card” or if something fails, the original “card” untouched. In addition, the issuer also get an attestation from the container if the process is successful which is quite important from a customer support point-of-view.
End-to-end security	☐	☑	KeyGen2/SKS depends on dynamically created and attested secret session keys which make issuers *de-facto talking directly to the container*. Most provisioning protocols do not give the issuer any indication of the container's identity or properties.
Algorithm agility	☐	☑	<keygen>: Only supports RSA keys. KeyGen2/SKS: Supports a negotiation method making it possible in a controlled fashion migrate to new/better algorithms. RSA, ECDSA/P-256, and SHA256 are mandatory.
Relative protocol complexity	1	25	KeyGen2 is language-wise about 25 times bigger than <keygen> but it also does at least 10 times as much. In spite of this complexity, the current proof-of-concept code and emulator show that the modular design of the protocol still makes implementations fairly simple and verifiable.
Relative system “fuzziness”	5	1	Although the SKS/KeyGen2 scheme is quite extensive, the fixed API and direct match between the protocol and the container makes it much easier to verify than a variant set of smart cards with highly operating-system dependent middleware. Device- and platform-independent JUnit test suites have been developed both for the KeyGen2 protocol and the low-level SKS API.
Implementation status	In most browsers except for MSIE	2011-04-30: SW emulator PoC	Microsoft's counterpart “CertEnroll” is considerably more sophisticated than <keygen> but is tightly bound to Windows as well as suffering from security issues making it difficult to deploy in a consumer environment.
Target audience	?	- On-line banking - e-Government solutions - Enterprise login - VPNs - Virtual credit cards - Virtual SIM cards - IdP login (InfoCards etc)	The current division between authentication solutions for different business segments appear to be mainly historical; Technically, a key is just a key.

Note: Describing the full SKS/KeyGen2 scheme in a table is not particularly realistic so this is just a short-list to make the objectives behind the effort a bit simpler to digest.

Version 0.12, Anders Rundgren, 2011-06-01

http://webpki.org/auth-token-4-the-cloud.html

http://www.w3.org/TR/html-markup/keygen.html