Profiles

- basic profile of XML Signature
  - flexible, but much easier to implement subset
  - ease more important than flexibility?
  - fundamentals to enable performance and interoperability
  - reduced attack surface
  - ease of understanding for implementers!
- deterministic-time processing
  - robustness against DoS
  - scaling behavior [~ O(n)]
- “bulk signing” -- relationship to non-XML-centric signing mechanisms

Profiles (II)

- use-case specific profiles – big picture?
- use-case-specific
  - implementations taught at run-time about constraints, turn off some features
  - run-time and implementation-time profiling
  - reuse of profiles?
  - WS-Policy, WS-SecurityPolicy?
  - protocol messages vs documents
- efficiently-verifiable (much easier to verify than to sign)
  - constraining transformations (XSLT)
  - discuss under “basic profile”

Refactoring? Backwards-compatibility?

- Existing specs?
  - WSS, XKMS, SAML
  - not being compatible means two stacks
- Predefined transformations to allow users to enforce profiles?
  - (-> transforms)
- longer-term view?

Refactoring? Backwards-compatibility?

- backwards-compatibility too constraining?
  - transition strategy: new applications MUST be able to verify old signatures – requirement for WG or for consumers?
  - not constrain in charter?
  - be explicit that 1.0 not deprecated
  - use extension points?
  - superset of old spec?
  - application vs infrastructure requirements
- impact of algorithm changes?
- efficient core + extensions?
  - leverage results from DSS?
- requirements work?
- **we get one breaking change.**

Implementation Guidance

- How to deal with it in scripting contexts?
- Order of operations
  - interaction with one-pass processing?
  - partitioning of attack surface anonymous vs non-anonymous
- feeds back to basic profile
- APIs
- how to really do “see what you sign”? -- best practices?
- wrapping countermeasures
- references moving around in documents
- improved security considerations

Key Management

- RetrievalMethod issue
- KeyInfo work more open than rest of spec
  - unclear to what extent spec is used
  - spec clarity
  - RetrievalMethod, X509data
  - naked keys (base64) vs self-signed certs
- XKMS for symmetric key cryptosystems
  - planning for a world post quantum computing?
  - building Kerberos-like systems on top of XKMS

Referencing Model

- processing vs. structural integrity protection
  - signature processing works fine
  - security properties aren't the ones you thought you got
- how to make ID-based approaches work?
  - talk about ID-ness without resorting to schema / xml:id
  - uniqueness when you have non-cooperating implementations (-> impl guidance?)
  - mutable IDs
  - communicate IDness as signature metadata? (can we flag ID attributes in context?)
  - reference mechanism for just one particular kind of ID?

Referencing Model

- Fully semantic, layered referencing
  - just not possible?
- structure-based approaches
  - use xpath for dereferencing instead of doing transformation?
- maybe something in the middle? combine?
  - remove features that get people in trouble?
  - or fall back to implementation guidance?

Algorithms

- NSA Cryptosuite B
- Randomization approaches
  - RSA-PSS
  - randomized hashing (RMX)
- Mandatory / optional to implement algorithms
  - what after SHA-1 (currently mandatory)
  - algorithms suites vs individual algorithms
- changing the defaults
  - define future algorithms in style used for RSA-PSS
- algorithm ID registry

Other issues

- Exclusive Canonicalization
- Schema extensibility / reuse of XML Signature
- XML 1.1
  - specs encouraged to move to XML 1.1 support
  - explore implications of XML 1.1 for new signature work
  - xpath 2.0 data model; distinct from mandating XPath 2.0
- encapsulating XML?

Transform model

- Use XProc to specify transformations, or roll out another one?
- How to apply XPath 1.0 to XML 1.1?
- Constrain transform model to enable pipelining with memory depending on document depth, not length
- Actual Implementations include optimized processing models for transform chains
  - interoperable subset?
- Effect on modularized c14n approaches?
- Standardized transforms for certain tasks?
- Require transforms to output documents instead of nodesets?

C14N overall

- not satisfactory
- dual purpose
  - output XML
  - process XML into an octet stream suitable for hashing
- simplifying current C14N?
- starting from scratch? (à la Thompson)
- splitting out certain features? layering? (Simon)
- relationship with EXI?
- integrating digest + c14n?
- output of transforms goes to applications AND combined “pre-hash”
- what does “what you sign is what you see” mean for c14n (and transforms)?
  - wrapping attacks

Interaction with other Groups

- XML Processing Model?
- ID-ness issue / how to benefit from xml:id
  - needs bigger picture
- Long-Term Archival
  - DSS
  - LTANS
- XADES @ ETSI
  - also long-term archival
- EXI
- customers: SAML, WS-I, WSS, Liberty (see current charter, plus some)
- DSS basic processing?

Requirements

- Performance