



Embedded security you can trust

# Standards Watch

Elliptic on security standards activity

Issue 3 – Winter 2006

## Welcome to Standards Watch

Welcome to the latest issue of Standards Watch. This issue digs deeper into the art and science of security standardization. The challenges in developing and implementing a standard are daunting and even the best struggle to get it right. This issue reviews key generation as standardized in ANSI X9.31 and the interpretation challenges encountered in what is now a venerable standard. The IEEE 1619 discussion continues as the standard is submitted for sponsor ballot and although the gap is closing there is still an issue that should be resolved before the standard goes mainstream. This is why Elliptic exists – not only to build rigorously tested solutions but also to help Elliptic customers successfully navigate the turbulent waters in both pre-standardization and post-standardization development scenarios.

### When a Standard's Not Standard

There are lots of jokes about standards and the standardization process. To paraphrase an old saw: "What's the definition of a perfect standard? Everyone's equally unhappy." But the best standards do something important for everyone: they make it easy for everyone to move on to the next big thing where they can add value. Standards are almost always a compromise, and everything is up for grabs in the compromise. WEP is an example of a standard that attempted to ignore the reality that wireless 802.11 networks would be very attractive targets to eavesdroppers. And not only to those who would be interested in free access to the Internet by stealing service paid for by others, but more insidiously to those intent upon stealing the information available in networks by using the weakness inherent in the standard to their advantage. This was (and continues to be) a tremendous black eye on the networking industry: not so much because the standard was a failure, but because the weaknesses were understood, documented and ignored long before the standard was adopted.

On the other hand, sometimes mistakes just happen. When they do, it is informative to see what happens in resolving them. The key generation algorithm of ANSI X9.31 is a case in point. For the most part, X9.31 is thought of as a complete, mature standard. However, there is at least one underlying issue that has become more relevant as time goes on. The problem relates to subtle differences between the normative text and the informative description regarding the generation of large prime numbers (hundreds of bits long) that are used

### In this Issue

A review of how difficult the standardization process is. We start with a retrospective view of ANSI X9.31 RSA key generation which offers a fascinating insight into how even a rigorous and mature standard can be subject to interpretation issues when rendered into practice by different implementers.

The first IEEE P1619 storage security standard is rapidly approaching ratification. And even this most carefully watched standards activity is threatened by interoperability concerns that should be addressed.

The lesson learned: standards evolve even after the balloting is done. Elliptic is here to help guide you throughout the entire process – from early stage negotiation of a new standard through implementation, testing and interoperability.

for asymmetric cryptographic keys. The normative text (standards-speak for the definition of a function or feature that can be validated in order to comply with the standard) provides a description of a process that accomplishes this. The normative text indicates that the implementer must: randomly guess a number, test each successive number to see if it is prime and return the prime number once found. The informative description provided in an annex to the specification amounts to: randomly guess a number, test it to see if it is prime and return it if it is, otherwise test each succeeding number until you find a prime. The difference is that the normative text amounts to a test for “greater than” versus a test for “greater than or equal to” in the informative description. In most cases, the results produced by the two versions are identical, but every few hundred trials, the results will be different since the initial random number guessed will actually be prime. So, the right approach to resolving this problem would be to ask the ANSI committee responsible for X9.31 to revise the standard to make the normative text and the informative description consistent. At the very least, it is clear that the normative text is the required functionality, and any validation of compliance to the specification should use the normative text.

Following publication of X9.31, NIST adopted it as a recommended method for generating secret keys, and there are NIST validation tests for compliance of implementations with X9.31. Unfortunately, the way both human nature and standards work is that most people take the path of least resistance. In practice implementers often use the non-normative examples since they provide convenient descriptions that map well onto real implementations. NIST's validation test unfortunately used the erroneous informative annex to develop the validation test code. Apparently many other implementations also used the same annex in their implementations since there are many approvals for compliant implementations under the NIST Cryptographic Algorithm Verification Program (CAVP).

As often happens in these situations, someone eventually discovered the discrepancy. This occurred because an implementation rigorously started from the beginning, using the normative definitions of the process to develop an implementation of the standard. Unfortunately when time came to validate the implementation, it failed on very rare occasions. In cryptography, this is very unusual: implementations usually work or they fail miserably. An investigation of the circumstances led to the realization that the validation code differed from the normative definition, the result being spurious failures. There were at least two reasonable responses to dealing with the discrepancy: change the normative text to agree with the reference implementation (apparently reasonable given how long it had taken to discover the discrepancy); or change the informative annex to agree with the normative text (the pedagogically correct, but apparently impractical response). There were also many less reasonable responses, and of course one of those was adopted: there are now two correct interpretations of the specification of this deterministic function that generate different, incompatible results. The current NIST interpretation of the specification is that one or the other of the ANSI definitions of the process may be used.

In many applications, this won't be a problem. After all, for many applications of RSA and related algorithms, the important point is that keys be large prime numbers chosen at random. Private keys generated according to either variant are both about as secure, and must remain secret in any case, so the discrepancy is minor and should be undetectable. However, there has emerged a new class of applications that do not perform a traditional key exchange, but rather use a deterministic process to re-generate keys at each of two (or perhaps

more) devices. (Microsoft uses this, for example, to allow users to move from one computer to another on a network without having to carry a device containing all their keys for different services they use with them. Details are disclosed in US patent 6,230,269.) For this to work, both sides of the link must use identical implementations. Given that the most standard implementations are those that claim compliance with ANSI X9.31, there is now a possibility that one side implements one X9.31 variant, while the other side implements the other variant. The result: randomly, they won't be able to communicate. This can end up being a serious impediment to adoption of this alternative approach to user authentication and privacy

A similar issue looms in the IEEE 1619 draft specification for disk encryption. Storage subsystem components have, over the years, been amongst the most interoperable computer devices known, and the SCSI interface standards have been a big part of the reason for this success. If a disk and the controller components share the same electrical interface, operation is pretty much plug-and-play: any disk can be connected to any electrically compatible controller and the disk (and more importantly the data on it) is immediately accessible.

Now along comes P1619. Its mandate is to “develop standards for cryptographic algorithms and methods for encrypting data before it is sent to the storage ... to create interoperable solutions.” At this writing (P1619 has adopted its draft 11 to be put out to sponsor ballot at IEEE), there is one obstacle to achieving this interoperability: a data object called a Data Unit Sequence Number (DUSN) has an unspecified encoding. The DUSN relates to the position of a data block on a disk (e.g. a sector number). By failing to specify an encoding, there is the very real possibility that disk and other storage subsystem controllers may each apply their own different encodings to the DUSN. If this happens it will lead to a situation in which the data on a disk can only be guaranteed to be successfully decrypted using the exact same controller model on which the data was originally written. This is clearly an undesirable situation, and flies in the face of interoperability. There is still time for the IEEE to resolve this before adoption of the standard and readers should pressure the IEEE for a decision prior to balloting.

## Quick Overview – IEEE P1619 Storage Security

IEEE 1619 defines “narrow block” encryption suitable for disk drives, meaning that the encrypted data that is written to disk has identical size to the unencrypted (plaintext) version of the same data. While this is desirable from a data management point of view, it is not ideal from a security point of view. The privacy of data stored

## Storage Encryption

StandardsWatch has looked at the IEEE 1619 standard for disk encryption in past issues, and it is discussed briefly elsewhere in this update. Storage security is an increasingly hot topic, and for all the right reasons. How many banks, on-line stores and health care providers need to lose the confidential financial and personal data of their customers before they take this issue of data security seriously? Or will legal standards, backed up by appropriate sanctions, be required to make those who collect and use this data behave responsibly. To date there has been precious little incentive aside from doing what is right. The result has been misplaced and stolen disks and backup tapes full of private data for thousands, or in some cases millions, of people. The IEEE P1619 family of standards is intended to provide the technical tools needed to address these problems. It remains to be seen whether the administrative and governance policy frameworks to apply those tools will also emerge.

on the disk in this way is protected, but unfortunately the integrity of that data is not guaranteed. Privacy is paramount to many users, so 1619 will be appropriate to those users.

However, some applications and users demand more from data protection. First, it is often desirable to be able to detect whether the data on disk has been altered (either accidentally or by an attacker) or moved from one location to another. For this reason 1619.1 defines “wide block” encryption, which adds cryptographic authentication to privacy. There's no free lunch, and in this case the price to be paid is expansion of the encrypted data compared with the plaintext data to provide storage for the cryptographic authenticator.

Similar in concept to 1619.1, 1619.2 takes this same approach and applies it in a form suitable for use with tape backup systems. The preferred implementation will be based on GCM-AES, which is suitable for high speed streaming encryption. Like the 1619.1 encryption, GCM-AES also provides authentication, protecting data written to tape from undetected modification.

Finally, the missing piece in all of the bulk data encryption standards has been key management. This is a huge issue in the storage context. In large data centers, it is essential to limit the lifetime of encryption keys, and manage keys to allow association of the correct keys with the data volumes to which they pertain. It is also necessary to specify how keys are protected, to provide for their secured backup and storage, to cycle keys over their lifetime, and to allow keys to be securely moved to the places they are needed. This will be the subject of IEEE 1619.3.

Over the next few issues of StandardsWatch the details of the various 1619 standards and the implications for implementers of secure storage systems will be examined in more detail.

#### Contact Information:

Elliptic  
308 Legget Drive, Suite 202  
Ottawa, ON K2K 1Y6  
+1-613-254-5456  
[www.ellipticsemi.com](http://www.ellipticsemi.com)  
Email: [info@ellipticsemi.com](mailto:info@ellipticsemi.com)