X9extra

Volume 2, Number 1
April 2011

Dear X9 Members:

When the word "encryption" comes up, most persons begin to squirm in anticipation of an unintelligible discussion on the subject. Many readers have little understanding of encryption other than the basic definition that encryption is a way to protect part of some message, and that to decode a message is to have the correct "key".

Well, this issue of X9 Extra is different in that the article by William Whyte, Security Innovation, offers a clear and understandable overview of encryption and how X9 standards change with the addition of new, more powerful encryption formats. The article also allows the reader to come away with a good understanding of how encryption works.

In the Financial Services Industry where handling personal data is a daily event, it is important to add encryption to protect the content of this information. In this environment it is important to use the most current form of encryption to ensure confidentiality.

With encryption developers looking for even tighter controls, safeguarding financial and other data continues to be an important activity of the X9 standards process. Please take the time to read this important document and gain more appreciation for the encryption process.

Sincerely,

Cindy
Cindy Fuller
Executive Director ASC-X9

X9.98: SECURE COMMUNICATIONS

Cryptographic Security for Financial Services

Financial institutions are increasing their electronic transactions, in both retail and back-office settings. As security depends more and more on cryptography, the X9F1 working group has been considering how to future-proof systems so that new cyber attacks, even previously unknown attacks, can be defeated. The new standard X9.98, "Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry," is one result of this process to provide a range of cryptographic choices suitable for the new threats of the 21st century.

X9.98

In November 2010, X9 issued X9.98, "Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry." This standard specifies the use of the NTRUEncrypt algorithm to establish secure communications for financial services. Although there are many existing X9 standards in the area of cryptography and secure communications, X9.98 marks a particularly significant step forward in improving the robustness of systems based on X9 standards: it allows the deployment of systems that are protected against quantum computing attacks as well as against classical attacks. This article explores the state of cryptography and why the new standard could be significant.

Cryptography: a (brief, recent) history

It was the middle of the 1960s, and James Ellis was unable to shake off a persistent problem and a persistent idea. Ellis was a cryptographer at the Communications Electronic Security Group (CESG), the secure communications arm of the British security services, tasked with breaking the communications of others and securing the communications of the UK and its allies. His problem had to do with key management. The present cryptographic algorithms were symmetric – and required sender and receiver to know/have the same key. This caused two big problems:

- First, in a large system of n devices, every device needed every other device's key – meaning there were n^2 keys in the system, but also making it very difficult to add new devices to the system; and

ACCREDITED STANDARDS COMMITTEE X9, INC.
1212 WEST STREET, SUITE 200, ANNAPOLIS, MD 21401
410.267.7707 WWW.X9.ORG

- Second, cryptography didn't just keep information secret (*encryption*): it also demonstrated the identity of the sender (*authentication*). This was vital in combat situations to run Identify Friend or Foe (IFF) protocols and make the split-second decision whether or not to fire.

But symmetric cryptography, based on shared secrets, meant that if someone compromised one unit of a group, they could potentially masquerade as any other unit. This meant they could infiltrate into combat groups and remain undetected for long enough to do damage. The only way around this problem was to have a single key for each pair of units, and that led straight back to the problem of distributing and managing an immense number of keys.

Those shortcomings were well known, and they were inherent in symmetric cryptography, and it was assumed that there was nothing a person could do about it. You couldn't make the sun rise in the west, and you couldn't do cryptography without a shared secret. But Ellis's persistent idea was "it didn't have to be that way". In 1970, in a paper that only a few hundred people could have read, he showed there could be a different kind of cryptography. He called it "non-secret cryptography", now better known as "public key" or "asymmetric" cryptography. With public key cryptography, a keyholder can prove their identity to anyone: you don't have to pre-distribute keys and you don't need a different key for every pair of units in the system. Introducing new units to the system becomes straightforward, and the problem of masquerading as other units goes away.

Ellis didn't have a method – an actual *algorithm* – for implementing non-secret cryptography, so his paper was regarded as a curiosity within the intelligence community. But in the next decade, his classified work was reproduced independently by academic researchers in the US, and these researchers did provide algorithms based on an area of mathematics known as *number theory*.

Whit Diffie and Martin Hellman invented a key agreement scheme based on modular arithmetic. Ron Rivest, Adi Shamir and Len Adelman invented RSA, an encryption and signature scheme based on factorization. When browser-based e-Commerce was first considered, RSA was a vital technology in making it possible. RSA allows a user, anywhere in the world, to go to their bank's web page and be sure (or nearly sure) that they are actually talking to their bank and that no-one else can read the messages they're sending. It was strong, it was standardized (in X9.31 and X9.44), and best of all – being built into the browser via the Secure Sockets Layer (SSL) protocol – it was free to the customer.

The future-proofing problem

However, there were two problems with RSA:

- First, it was slow – not on the browser side, but on the server side, where the mathematically complex private key operations were carried out; and
- Second, it was the only one.

What if an individual RSA scheme wasn't secure? When RSA was first published in 1977, Rivest casually stated that it would take "millions of years" to break keys that were 512 bits long (about 51 decimal digits). In fact, a 512-bit key was broken in 1999. The difference was partly because Rivest had miscalculated in the first place, but partly because breaking techniques improved. What if they improved the scheme further? The risk of monocultures is that a single disease can wipe them out. RSA was a monoculture, and the cost of a break would be to leave consumer transactions worldwide open to attack.

Academics continued to work on developing alternative public key cryptosystems and discovered something uncomfortable: apparently very different cryptosystems could be vulnerable to the same attacks. Diffie-Hellman (and the associated signature algorithm, DSA) appeared to be based on a different hard problem from RSA, but it turned out that almost identical techniques would break both. Mathematicians cast the net wider and, in 1985, Neal Koblitz and Victor Miller proposed the use of Elliptic

We Want You!

You like to put your thoughts into words, and you like to express your ideas to others. You take pride in your writing talent, and you know how to get your thoughts across to your reading audience. You can project your thoughts in a deep technical and elevated level. Actually, you think of yourself more as an author, not just a technical writer.

If this sounds like you, why not contribute your skills and knowledge to writing an article for the X9 EXTRA. You could be helping your fellow X9ers learn something they perhaps never knew or realized about our community and standards development.

If this peaks your interest, then please contact Brian Salway bsalway@symcor.com or Ed Stana ed.stana@x9.org of the ASC X9 Marketing and Membership Committee. You don't need to be a professional writer, you just require the knack for written communications and you need to know your subject matter. Above all, you're interested in doing something perhaps different and challenging for X9. Thanks for your help!

—X9's "All Committee Meetings" scheduled for San Antonio, Oct. 24–28, 2011—

Curve Cryptography (ECC). ECC has many advantages over RSA: it's smaller, it's faster (on the server) and, crucially, it depends on different mathematics. The best known attacks against RSA and Diffie-Hellman simply do not work against ECC. ECC is becoming a widespread backup algorithm, and in some settings (such as particularly constrained devices) is the cryptographic algorithm of choice. It has been standardized by X9, in X9.62 and X9.63 for signatures and key establishment respectively.

Quantum computing: a new threat

Progress in cryptanalysis continues, and the 1990s saw a new paradigm: quantum computing. Quantum computers operate on quantum bits or qubits, which unlike classical bits can be both 1 and 0 at the same time. In theory, this lets them carry out an enormous number of calculations – 20 qubits can represent a million different states, and the quantum computer can find which state best meets a set of conditions by calculating on all the states simultaneously. There are enormous technical difficulties to developing a working quantum computer, but progress is being made rapidly. However, even when quantum computers are built, there's a problem that significantly reduces their power – getting the answer back out. To extract information from a quantum computer requires a series of small, reversible steps that exactly preserve the energy levels within the computer. If those steps are carried out wrong, the quantum state is broken and the quantum information is lost. This isn't just an implementation issue – it is fundamental to quantum computers themselves.

Much ingenuity has been put into developing algorithms that allow rapid extraction of quantum information, but to date with mixed success. There is one – Peter Shor's algorithm, published in 1994. Unfortunately for cryptography, this single standout can be used to break all the existing, standardized public key

algorithms: RSA, Diffie-Hellman, and ECC. Shor's algorithm is so devastating that the usual approach to improving public key cryptography – increasing the key length – will not work in this case: when quantum computers are developed, no RSA, Diffie-Hellman, or ECC key can be made safe.

Since Shor's algorithm was published, researchers in the US and internationally, have tried many different approaches to developing quantum computers. The clock has been ticking for existing public key algorithms. But no one knows how fast it's ticking. All approaches to date have run into problems with scaling – approaches that work for 5 or 7 qubits can't be made to work for larger systems. But at some point, with high probability, a research team will come up with an approach that works, and the financial services community must be prepared when this happens.

Future proof cryptography: X9.98

X9.98 is based on the NTRUEncrypt algorithm, first published in 1998. NTRUEncrypt (originally just called "NTRU") was developed by Jeff Hoffstein, Jill Pipher and Joe Silverman, mathematicians at Brown University in Providence, RI. NTRU has two advantages:

1) It is extremely fast – up to 60 times faster than RSA for private key operations; and
2) It is based on entirely different math from RSA and ECC and no known quantum computing algorithm exists to break it

Given these advantages, the X9F1 group decided in 2003 to start creating a standard based on NTRUEncrypt, which was referred to as "Lattice Based Polynomial Public Key Encryption" (LBP-PKE) to avoid the use of trademarks owned by NTRU Cryptosystems, Inc.

Moving forward with this standard involved balancing several considerations. X9F1 does not want to standardize new cryptographic algorithms indiscriminately, as new standards potentially create a new burden of support for X9's customers. The more algorithms an implementer needs to implement, the greater the implementation complexity and the higher risk that the implementation will contain flaws. Also, even if new algorithms are believed to be resistant to quantum computers, the fact that they are relatively new means that they have not had the time to have the same level of analysis as established algorithms and they may in fact be vulnerable to undiscovered classical as well as quantum attacks. The working group moved very carefully through the project, ensuring at every step that there was still support for completion, and as a result, the standardization process took longer for X9.98 than is usually the case.

For example, the committee took the unusual step of soliciting input from external cryptographic experts to determine the opinion of the cryptographic community at large about the security claims for LBP-PKE. These experts (including the European ECRYPT expert group) reported back that the security estimates were credible and there was no barrier to standardization, but emphasized that new algorithms may be vulnerable to as yet undiscovered attacks – but so, of course, can existing algorithms.

Also, during the standardization process, additional improvements to attacks against LBP-PKE were in fact discovered. However, these were clearly just improvements to attacks, not fundamental new approaches, and the working group was satisfied that the improvements did not constitute a reason to fear that there were fundamentally new attacks. The final standard contains text alerting implementers to the fact that LBP-PKE is more recent than other algorithms and noting that implementers may choose to take that into account when deciding which public key algorithm to implement.

How does Lattice Based Polynomial Public Key Encryption work?

The cryptography standardized in X9.98 is based on the well-known closest lattice vector problem. In the diagram below, the lattice is the set of intersections of the blue lines: every point can be expressed as an integer times (f,g) (= (7, 1)) plus an integer times (F,G) (= (2, 18)). But all of these points can also be expressed as an integer times (1, h) (= (1, 71)) plus an integer times (0, q) (= (0, 124)). The vectors ((f,g), (F, G)) are a basis for the lattice, and ((1, h), (0, q)) are also a basis.

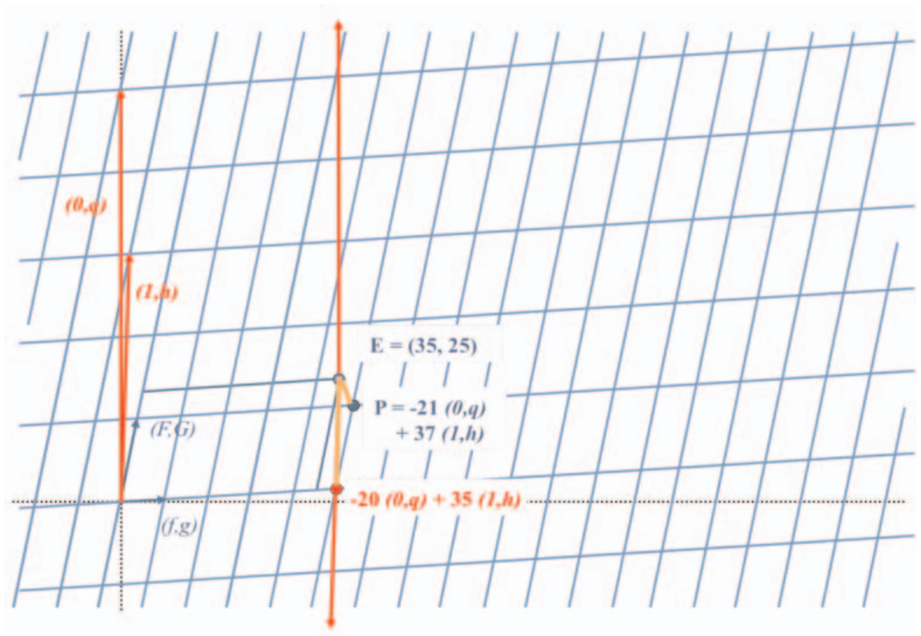
Not every point in space is a point in the lattice. The point E = (35, 25) is not in the lattice, but anyone who knows a basis can use it to find the closest lattice point. An easy way to do this is express E as a non-integer combination of the basis vectors, and simply round. So

E = 4.6 (f, g) + 1.3 (F, G),

giving the closest point P = 5(f, g) + 1 (F, G) = (37, 23). However,

E = 35(1,h) - 19.84 (0, q),

giving the closest point (35, 5). The basis with longer vectors gives a "closest point" that is considerably further away. We call the basis with longer vectors a "bad basis" and the basis with shorter vectors a "good basis". In two dimensions, it's easy to convert from a good basis to a bad basis, but in large numbers



of dimensions (and LBP-PKE typically uses over 800 dimensions) this is a very hard problem. If the encrypted message is a point in space, and the secret message is a lattice point nearby, the good basis is a private key that lets the holder decrypt, and the bad basis is a public key that allows encryption but doesn't give the holder any significant ability to decrypt other cipher texts.

Lattice-based algorithms have been known since the early 1990s and are the subject of vigorous research. They are widely believed to be secure for appropriate parameter choices. NTRUEncrypt / LBP-PKE is unique among lattice based algorithms in having a relatively small key size, as well as the efficient operations and quantum computing resistance that are common to all lattice-based cryptosystems.

The table below shows the relative performance of LBP-PKE, RSA, and ECC.

Conclusion

In developing X9.98, the X9F1 working group is continuing to prudently extend the range of cryptographic algorithms available to implementers in the financial services industry. Cryptographers may choose to implement LBP-PKE on its own, to take advantage of the speed benefits, or in combination with an existing public key algorithm, to ensure that their systems are quantum-proof while retaining the security properties of their existing systems. Quantum computers may be a long way off, but X9.98 gives the financial services providers some of the tools they will need to cope with this distant but real threat.

Relative Performance of LBP-PKE, RSA, and ECC

Security Level (bits)	LBP-PKE Key size	ECC Key Size	RSA Key Size	LBP-PKE ops/ Sec	ECC ops/ Sec	RSA private key ops/ Sec
112	5951	224	2048	2284	951	156
128	6743	256	4096	1896	650	12
192	9757	384	7680	1034	285	8
256	12881	512	15360	638	116	1