Lecture 14: Elliptic Curve Cryptography and Digital Rights Management

Lecture Notes on "Computer and Network Security"

by Avi Kak (kak@purdue.edu)

May 27, 2017
9:52pm

Goals:

- Introduction to elliptic curves
- A group structure imposed on the points on an elliptic curve
- Geometric and algebraic interpretations of the group operator
- Elliptic curves on prime finite fields
- **Perl and Python implementations for elliptic curves on prime finite fields**
- Elliptic curves on Galois fields
- Elliptic curve cryptography (EC Diffie-Hellman, EC Digital Signature Algorithm)
- Security of Elliptic Curve Cryptography
- ECC for Digital Rights Management (DRM)

CONTENTS

14.1: WHY ELLIPTIC CURVE CRYPTOGRAPHY?

- As you saw in Section 12.12 of Lecture 12, the computational overhead of the RSA-based approach to public-key cryptography increases with the size of the keys. As algorithms for integer factorization have become more and more efficient, the RSA based methods have had to resort to longer and longer keys.

- Elliptic curve cryptography (ECC) can provide the same level and type of security as RSA (or Diffie-Hellman as used in the manner described in Section 13.5 of Lecture 13) **but with much shorter keys**.

- Table 1 compares the key sizes for three different approaches to encryption for comparable levels of security against brute-force attacks. What makes this table all the more significant is that for comparable key lengths the *computational burdens* of RSA and ECC are comparable. *What that implies is that, with ECC, it takes one-sixth the computational effort to provide the same level of cryptographic security that you get with 1024-bit RSA.*

  [The table shown here is basically the same table as presented earlier in Section 12.12 of Lecture 12,

except that now we also include ECC in our comparison.] [In case the reader is wondering why we placed the word *key* between quotation marks in the header of the "RSA and Diffie-Hellman" column in Table 1, strictly speaking what is being referred to there is the size of the modulus. (Note however that in most cases the size of the private key is comparable to the size of the modulus.) The reason for quoting *key* in the header for the ECC column is the same, as you will see in this lecture.]

Symmetric Encryption Key Size in bits	*RSA and Diffie-Hellman "Key" size in bits*	*ECC "Key" Size in bits*
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 1: *A comparison of key sizes needed to achieve equivalent level of security with three different methods.*

- The computational overhead of both RSA and ECC grows as $O(N^3)$ where N is the key length in bits. [Source: Hank van Tilborg, NAW, 2001] Nonetheless, despite this parity in the dependence of the computational effort on key size, it takes far less computational overhead to use ECC on account of the fact that you can get away with much shorter keys.

- Because of the much smaller key sizes involved, ECC algorithms

can be implemented on **smartcards** without mathematical co-processors. **Contactless smart cards** work only with ECC because other systems require too much induction energy. Since shorter key lengths translate into faster handshaking protocols, ECC is also becoming increasingly important for **wireless communications**. [Source: Hank van Tilborg, NAW, 2001]

- For the same reasons as listed above, we can also expect ECC to become important for **wireless sensor networks**.

- If you want to combine forward secrecy, in the sense defined in Section 12.6 of Lecture 12, with authentication, a commonly used algorithm today is ECDHE-RSA. [The acronym "ECDHE" stands for "Elliptic Curve Diffie-Hellman Ephemeral". You will also see in common use a variant acronym: ECDH-RSA. The difference between ECDHE and ECDH is that the "ephemeral" implied by the last letter in the former implies just a one-time use of the session key.] In ECDHE-RSA, RSA is used for certificate based authentication using the TLS/SSL protocol and ECDHE used for creating a one-time session key using the method described in Section 14.12. [You could also use DHE-RSA, which uses the regular Diffie-Hellman Exchange protocol of Section 13.5 of Lecture 13 for creating session keys, for the same purpose. However, you are likely to get greater security with ECDHE-RSA.] [The main reason RSA is widely used for authentication is because a majority of the certificates in use today are based on RSA public keys. However, that is changing. You now see more and more organizations using ECC based certificates. ECC based certificates use the ECDSA algorithm for authentication. This algorithm is presented briefly in Section 14.13. When authentication is carried out with ECDSA and the session key generated with ECDH or ECDHE, the combined algorithm is denoted ECDHE-ECDSA