The Factoring Dead
Preparing for the Cryptopocalypse

Thomas Ptacek, Matasano
Tom Ritter, iSEC Partners
Javed Samuel, iSEC Partners
Alex Stamos, Artemis Internet
Agenda

1. Introduction
2. The Math
3. New Advances
4. The Impact
Why are we here?

• There is a significant disconnect between theory and reality in security.
  • Lots of great, continuous academic research in cryptography.
  • Few engineers get beyond Applied Cryptography before shipping code.
  • In the 2010's, it is no longer acceptable to just use standard libraries and claim ignorance.

• We wanted to see if we could bridge this gap a bit.

• We certainly are not the only ones to do so.
Recent TLS Problems

• Numerous attacks on the current TLS infrastructure.
  • BEAST \(^1\)
  • CRIME \(^2\)
  • Lucky 13 \(^3\)
  • RC4 Bias \(^4\)
• Even a new compression oracle attack here at BlackHat USA 2013! \(^5\)
• Were any of these attacks really unpredictable to people paying attention? (Hint: no\(^6\))

Comparison to Academic Time Line

- 1998 – EFF Deep Crack defeats DES in 56 hours
- 2005 - Pre-image attacks against MD5 discussed
- 2008 - Applebaum, Sotirov et. al. use MD5 attack against CA
- 2011 - CA/Browser Forum forbids MD5
- 2012 - Somebody (cough) uses related attack against Microsoft for FLAME
- SIM Card Attack at BlackHat 2013 using DES
Why such a disconnect?

- Most systems are not designed for cryptographic agility
- Cryptography is an ecosystem
- Few companies employ full-time cryptographers
- Hard for InfoSec practitioners to keep up-to-speed
- Lots of momentum in the professional consulting core.

*We have failed as an industry to address these structural problems.*
Why are we here?

• Looking for the next crypto black swan.
• Our thesis:
  • Last six months has seen huge leaps in solving the DLP
  • These leaps have parallels to the past.
  • There is a small but real chance that both RSA and non-ECC DH will soon become unusable.
  • Ecosystem currently cannot support a quick pivot to ECC

We want this room to become the seed of change
Why Asymmetric Cryptography?

- Key part of modern cryptosystems
How does asymmetric crypto work?

• We need a “trap-door” function, something that is easy to do but hard to undo
  • We also need a way to cheat with more information

• Rarely is the difficulty of this function proved, only assumed
What are the common primitives?

- **Diffie-Hellman** - 1976 - Secure key exchange

- **RSA** - 1977 - Encryption, signing

- **Elliptic Curve Cryptography**
  - Suite B - 2007 - Key exchange, signing and encryption
  - GOST - 2010 - Key exchange, signing and encryption
First published by Whitfield Diffie and Martin Hellman in 1976

Establishes shared secret by exchanging data over a public network.

Security relies on the hardness of the discrete logarithm problem.
How do I attack DH?

• Solve the discrete logarithm problem:
  • Suppose \( h = g^x \) for some \( g \) in the finite field and secret integer \( x \).
  • The discrete logarithm problem is to find the element \( x \), when only \( g \) and \( h \) are known.

• Also how you attack El-Gamal and DSA
RSA Overview

• Key Generation to compute public and private key exponent \((e, d)\)

• Encryption by raising the message to public key exponent \(e\)

• Decryption by raising the message to private key \(d\)

• Security relies on the hardness of factoring.
How do I attack RSA?

- Factoring!
  - Find the $p$ & $q$ such that $p\times q = N$

- Factoring an RSA modulus allows an attacker to compute the secret $d$ and thus figure out the private key.
Elliptic Curve Cryptography Overview

- An elliptic curve $E$ over $R$ real numbers is defined by a Weierstrass equation eg $y^2 = x^3 - 3x + 5$

- Cryptographic schemes require fast and accurate arithmetic and use one of the following elliptic fields.
  - Prime Field $F_p$ where $p$ is a prime for software applications.
  - Binary Field $F_{2^m}$ where $m$ is a positive integer for hardware applications.
• ECC is secure due to the hardness of the elliptic curve discrete logarithm problem (ECDLP).
  • Given an elliptic curve $E$ defined over a finite field $F_{q^l}$, a point $P \in E(F_{q^l})$ of order $n$, and a point $Q \in E$
  • Find the integer $d \in [0; n - 1]$ such that $Q = dP$
## ECC v RSA key sizes

<table>
<thead>
<tr>
<th>Symmetric</th>
<th>DH or RSA</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>512</td>
<td>112</td>
</tr>
<tr>
<td>80</td>
<td>1024</td>
<td>160</td>
</tr>
<tr>
<td>112</td>
<td>2048</td>
<td>224</td>
</tr>
<tr>
<td>128</td>
<td>3072</td>
<td>256</td>
</tr>
<tr>
<td>192</td>
<td>7680</td>
<td>384</td>
</tr>
<tr>
<td>256</td>
<td>15360</td>
<td>521</td>
</tr>
</tbody>
</table>

### NIST Recommended Key Sizes
Agenda

Introduction

The Math

New Advances

The Impact
Discrete Logarithm Algorithms

• Generic algorithms (for any G)
  • Example: Pohlig-Hellman
  • Shows that discrete logarithm can be solved by breaking up the groups into subgroups of prime order.
  • Generic algorithms are exponential time algorithms.

• Specific algorithms which make use of group representation
  • Example: Index calculus algorithms
  • They leverage particular properties of the group
  • Result in sub-exponential running time
Exponential vs Polynomial

L(0) – Polynomial
Fast enough to scare you

L(1) – Exponential
Way Too Slow

Linear running time plot
Logarithmic running time plot
Exponential vs Polynomial

$L(0)$

$L(1/2) – 1979$

$L(1)$ — current fastest ECDLP algorithms
Exponential vs Polynomial

\[ L(0) \]

\[ L(1/2) - 1979 \]

\[ L(1/3) - 1984 \]

Factoring and Discrete Logs stay here for the next 30 years

\[ L(1) - \text{current fastest ECDLP algorithms} \]
Exponential vs Polynomial

L(0)

L(1/3) – 1984
Factoring

L(1/2) – 1979

L(1) – current fastest ECCDLP algorithms

L(1/4) for Discrete Logs with restrictions on the types of group - 2013
Exponential vs Polynomial

L(0) - current fastest ECC algorithms
L(1/2) – 1979
L(1/3) – 1984
Factoring
L(1/4) - 2013
L(0) for discrete logs with restrictions on the types of groups – 2013
New Developments in 2013

• Rapid progress in DL research in past 6 months
  • February 20, 2013: Joux published a $L(1/4)$ algorithm to solve DLP in small characteristic fields.
  • April 6, 2013: Barbulescu et al solve the DLP in of $\mathbb{F}_2^{809}$ using the Function Field Sieve algorithm (FFS)
  • June 18, 2013: Barbulescu, Gaudry, Joux, Thomé publish a quasi-polynomial algorithm for DLP in finite fields of small characteristic.
Joux’s New Discrete Log Algorithm (Feb 2013)

- Uses judicious change of variables to find multiplicative relations easier.

- Uses a specific polynomial with linear factors to simplify the computation.

- Uses a new descent algorithm to express arbitrary elements in the finite field.

- Complexity is $L(1/4 + o(1))$ which is considerably faster than any discrete logarithm algorithm published before.
More Improvements
June 2013, Barbulescu, Gaudry, Joux, Thomé

• **Quasi-polynomial algorithm** for DL in finite fields of small characteristic.

• Improves Joux’s February 2013 algorithm using special matrix properties.

• Fastest discrete logarithm has been improved significantly in the past 6 months after marginal progress in 25 years.

• However; no clear jump to more practical implementations which use finite fields with larger characteristic YET!
Implications of Discrete Log Progress

• Pairing based cryptography (PBC) over small characteristics is no longer secure.
  • PBC can be used for identity-based encryption, keyword searchable encryption where traditional public key cryptography may be unsuitable.
  • Currently used mainly in academic circle.

• Improves the Function Field Sieve (FFS) in most cases.
  • The function field sieve currently can be used to solve for small to medium characteristics fields.
Why Should I Care?
Function Field Sieve

• Function Field Sieve has Four Steps
  • Choose a Polynomial
  • Relation Filtering
  • Linear Algebra
  • The Descent

• In the last 6 months, all of them have been improved
  • More likely something can be used on something we care about

• His record setting calculation, in May, took 550 Hours
  • 512 Bit RSA takes 652 Hours
Attacking DH, DSA, ElGamal

• Joux has attacked fields of a small characteristic
• We use fields of a large characteristic

• Joux’s...
  • Polynomial choice probably would not help
  • Sieving Improvements may help
  • Descent Algorithm needs tweaking, but definitely helps

• Renewed interest could result in further improvements.
Attacking RSA

• Factoring advances tend to lead to advances in Discrete Log

• Discrete Log advances tend to lead to advances in Factoring

• Degrees of difficulty of both problems are closely linked.
Mutual Advances over the years


• 1984 Quadratic Sieve Factoring -> 1987 improvements in Discrete Log Index Calculus Algorithms.

• 1993/4 Discrete Log Number & Function Field Sieves -> 1994 General Number Field Sieve for Factoring.
Factoring vs Discrete Logs

Factoring
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. Square Root

Discrete Logs
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. The Descent
Factoring vs Discrete Logs

Factoring
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. Square Root

Discrete Logs
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. The Descent

Not That Slow
Constant Time

Not That Slow
Constant Time

Not That Slow
Constant Time

Not That Slow
Constant Time
Factoring vs Discrete Logs

Factoring
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. Square Root

Discrete Logs
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. The Descent

Not That Slow
Constant Time
Easy to Parallelize

Not That Slow
Factoring vs Discrete Logs

Factoring:
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. Square Root

Discrete Logs:
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. The Descent

- Not That Slow
- Constant Time
- Easy to Parallelize
- Slow & Difficult to Parallelize
Factoring vs Discrete Logs

Factoring
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. Square Root

Discrete Logs
1. Polynomial Selection
2. Sieving
3. Linear Algebra
4. The Descent

- Not That Slow
- Constant Time
- Fast
- Slow & Difficult to Parallelize
- Easy to Parallelize
- Very Slow
- Very Fast
Attacking RSA

• No obvious technique right now from Joux’s improved discrete logarithm algorithm that applies directly to factoring.

• But I’m not a mathematician, I just play one on stage – I wouldn’t bet the farm on that

• Public colloquium and publications seem to indicate that NSA/NIST may also already be very concerned.
Public Implementations & Tutorials

• MSIEVE
  • http://sourceforge.net/projects/msieve/

• CADO-NFS
  • http://cado-nfs.gforge.inria.fr/

• GGNFS
  • http://www.math.ttu.edu/~cmonico/software/ggnfs/

• Tutorials
  • http://github.com/tomrittervg/cloud-and-control
Implications

• ECC is still standing - still requires exponential time algorithms

• If Joux or others hits upon a general purpose discrete logarithm algorithm as fast his special purpose one...
  • Diffie-Hellman, DSA, and El-Gamal are toast
  • If that leaps to factoring - RSA is toast

• Technically not dead, but...
  • RSA key sizes may have to go up to 16,384 bits
  • Wildly impractical for actual use, never mind that nothing supports keysizes that large
Agenda

Introduction

The Math

New Advances

The Impact
What Happens If DH or RSA Fails Now?

• Widespread active and passive attacks against live and recorded TLS.
  • PFS not necessarily the panacea

• Failure of code-signing and update mechanisms
  • How do you fix your software

• Failure of PGP, S/MIME and most end-to-end encryption

• Almost total failure of trust in the Internet
So, what now?

• We need to move to ECC, rather quickly

• Alex says that ECC is perfectly secure, YAY!

• Not really
  • <30 years of research versus 400:
  • Uses some of the same ideas

• Right now it’s all we have
  • Long-term, we need more research into alternatives
  • RSA was 1977, RC4 was 1984. Give Rivest a break.
Why has ECC uptake been so slow?

- Lots of push from academia and government into ECC
- DH/RSA are here and they are easily understood
- Legal risks have slowed ECC adoption
- ECC had compatibility problems, but NIST has specified 15 standard curves
Overview of Suite B

• In 2005, the NSA released the Suite B set of interoperable standards

• Suite B specifies:
  • The encryption algorithm (AES-256)
  • The key exchange algorithm (Elliptic Curve DH)
  • The digital signature algorithm (Elliptic Curve DSA)
  • The hashing algorithms (SHA-256 and SHA-384)

Hmm, what’s missing?
ECC Patents

• The patent issue for elliptic curve cryptosystems is the opposite of that for RSA and Diffie-Hellman.
  • RSA and Diffie-Hellman had patents for the cryptosystems but not the implementation.

• Several important ECC patents owned by Certicom (Blackberry)
  • Efficient $GF(2^n)$ multiplication in normal basis representation.
  • Technique of validating key exchange messages to prevent a man-in-the-middle attack.
  • Technique for compressing elliptic curve point representations.
• NSA purchased from Certicom (now Blackberry) a license that covers all of their intellectual property in a restricted field of use.

• License is limited to implementations that were for national security uses and certified under FIPS 140-2 or were approved by NSA.

• Commercial vendors may receive a license from NSA provided their products fit within the field of use of NSA’s license.

• Commercial vendors may contact Blackberry for a license for the same 26 patents.
Maybe Certicom is cool about this?
# ECC Support on Operating Systems

<table>
<thead>
<tr>
<th>OS</th>
<th>Library</th>
<th>ECDH</th>
<th>ECDSA</th>
<th>Others</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSX/IOS</td>
<td>ssl-36800</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>10.6</td>
</tr>
<tr>
<td>OSX/IOS</td>
<td>smime-36873</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>10.6</td>
</tr>
<tr>
<td>Windows</td>
<td>CNG</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>Vista</td>
</tr>
<tr>
<td>Windows</td>
<td>TLS</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>Vista</td>
</tr>
<tr>
<td>Windows</td>
<td>Suite B</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>Vista SP1, Windows 7</td>
</tr>
</tbody>
</table>

Table: Windows and OSX ECC Support
# ECC Support on Android

<table>
<thead>
<tr>
<th>OS</th>
<th>Library</th>
<th>ECDH</th>
<th>ECDSA</th>
<th>Others</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>Bouncy Castle</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>4.0</td>
</tr>
<tr>
<td>Android</td>
<td>TLS</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>3.2.4</td>
</tr>
<tr>
<td>Android</td>
<td>CyaSSL</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>2.4.6</td>
</tr>
<tr>
<td>Android</td>
<td>NSS</td>
<td>Yes</td>
<td>Yes</td>
<td>NTRU</td>
<td>3.11</td>
</tr>
</tbody>
</table>

Android ECC Support
### ECC Support on Programming Languages

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>Library</th>
<th>ECDH</th>
<th>ECDSA</th>
<th>Others</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>PyECC</td>
<td>Yes</td>
<td>Yes</td>
<td>ECIES</td>
<td>2.4</td>
</tr>
<tr>
<td>C</td>
<td>OpenSSL</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>3.2.4</td>
</tr>
<tr>
<td>Java SE6</td>
<td>Bouncy Castle</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>Java 6</td>
</tr>
<tr>
<td>Java SE7</td>
<td>Native</td>
<td>Yes</td>
<td>Yes</td>
<td>ECIES, ECDSA, ECHR</td>
<td>Java 7</td>
</tr>
<tr>
<td>Ruby</td>
<td>OpenSSL</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Programming Languages ECC Support
Code Signing

• Windows Code Signing
  • Default is RSA
  • ECC is supported through CSPs but not default

• Android Code Signing
  • Both DSA and RSA are currently supported.

• iOS code Signing
  • Uses CMS
  • Supports ECDH and ECDSA.
Transport Encryption

• TLSv1.2 is the first to include ECC options
  • Only TLS_RSA_WITH_AES_128_CBC_SHA is required

• Before TLS 1.2, CA and Cert had to match.
  • With 1.2 you can cross-sign
  • Can use DH_DSS, DH_RSA, ECDH_ECDSA, and ECDH_RSA with either ECC or RSA

• TLS 1.1 supports ECDH(E) for PFS
ECC roots exist, buying a cert is not so easy

There would significant work required in the transition form RSA to ECC certificates.

- **Thawte Root Certificate** - Root CA is not used today. Intended for use in the future for SSL certificates.
- **Verisign/Symantec Root Certificate** - ECC root certificate for 5 years; just begun offering commercial certificate this year.
- **Entrust ECC Certificate** - No global root certificate currently available today. Will use a Public ECC-256 Root.
- **Comodo** - 384 bit ECC Root certificate.
• Current Root KSK generated in 2010 (algorithm 8)
  • Standard specifies rotated “when necessary” or at five years
• IANA, Verisign, ICANN SSAC looking at options
• ECC being considered
  • Helps with Zone File size
• Interesting enough, check out .ru
Other Popular Applications

- BlackBerry uses ECC extensively

- OpenVPN uses OpenSSL which includes ECC support, doesn’t seem to work

- IPSEC - Cisco, Shiva and Nortel gateways support ECDH IKE.

- OpenSSH has ECC support, not the default.
What do you do now?
If you are a... OS or language vendor

• Make ECC easy to use
  • See NaCl’s box() and unbox()

• Update documentation to push developers away from RSA

• Get aggressive about compatibility testing

• Eat your own dogfood
If you are... a browser vendor

- TLS 1.2 needs to be a P1 feature
  - Only IE 11 and Chrome 29 support (both pre-release)

- Push at CA/B Forum for standardized process for cross-signed certificates
If you are a software maker

• You need to support TLS 1.2 on endpoints

• Build systems with pluggable primitives
  • Versioning
  • Handshake and negotiation
  • If this sounds too hard use TLS 1.2

• Use ECC for any new cryptosystems

• Retrofit old mechanisms using wrapping
  • ECC signed binary inside of legacy RSA signature
If you are a... Certificate Authority

• Make it easy to buy an ECC cert

• Change documentation to include ECC CSR instructions

• The CA/Browser Forum should promulgate standards pushing this
If you are... BlackBerry

• Make the world a safer place...

• License the ECC patents openly to any implementation of Suite B, regardless of use
If you are... just a normal company

• Use ECC certificates where possible
• Bug vendors for TLS 1.2 and ECC support
• Turn on ECDHE PFS today!
• Survey your exposure, so when the cryptopocalypse comes you are like this guy:
Summary

• Current cryptosystems depend on discrete logarithm and factoring which has seen some major new developments in the past 6 months.

• We need to move to stronger cryptosystems that leverage more difficult mathematical problems such as ECC.

• There is a huge amount of work to be done, so please get started now.
Thank You

- JasonP
- Antonie Joux
- Dan Boneh
- Ryan Winkelmaier (iSEC Partners intern)