Dmytro Sotnyk

Hadoop Security Project

Asymmetric Encryption Algorithms overview

2016

Goal of this investigation

In this investigation we will be focused on RSA and ECC asymmetric encryption algorithms.

We will overview and estimate:
- known vulnerabilities of RSA and ECC algorithms
- encrypted/unencrypted data size ratio
- performance of encryption/decryption for chosen algorithm
- dependency between key length, security and performance, if any

Also we will overview hybrid cryptosystems.

We excluding EPKE (Enveloped Public Key Encryption) from our investigation and remain focused on PKE only.

Prerequisites

To understand and verify this investigation, reader need to overview next articles in advance

Key Distribution
Diffie–Hellman key exchange
Diffie-Hellman Example
Diffie–Hellman problem
ElGamal encryption
Adventures in Encryption : ElGamal Encryption
ElGamal example
ElGamal signature scheme
DSA
Public-key cryptography
Hybrid cryptosystem
Elliptic Curve Digital Signature Algorithm
GNU Privacy Guard
RFC4880: OpenPGP Message Format
Pretty Good Privacy
PKCS 1
PKCS
RSA
Chosen-ciphertext attack
Optimal asymmetric encryption padding

Why RSA encryption padding is critical
Coppersmith's Attack
Wiener's attack
Cryptanalytic Attacks on RSA
A Survey of Cryptanalytic Attacks on RSA
Transport Layer Security
Elliptic curve cryptography
RFC6637 : Elliptic Curve Cryptography (ECC) in OpenPGP
Elliptic Curve Cryptography in Practice
OpenSSL : Elliptic Curve Cryptography
The RSA Algorithm
ECC vs RSA: Battle of the Crypto-Ninjas

Otherwise, see Bottomline section

RSA Known Weakness

Mathematical weakness

In general, known attacks on RSA described here

RSA
Chosen-ciphertext attack
Coppersmith's Attack
Wiener's attack
Cryptanalytic Attacks on RSA
A Survey of Cryptanalytic Attacks on RSA

The Department of Defense spends $11 billion a year on cryptanalysis, employing some 35,000 people full time to the task, but results (if they are) are classified.

There is no known practical attacks except chosen-ciphertext attack. So proper implementation of padding need to be used, see Optimal asymmetric encryption padding and PKCS 1.

Implementation weakness

Since theoretically RSA is strong (or declared strong) enough and governments are interested in crypto breaking, RSA crypto was seriously affected by different artificial vulnerabilities specially added to implementations.

Such information is mostly classified and we have no goal to make information secure from the government. Nevertheless, we have to understand, are these artificial vulnerabilities can be used by a hackers and, if yes, how to protect data.

Backdoor in random generator

Accordingly to Reuters ("Exclusive: Secret contract tied NSA and security industry pioneer"), NSA added backdoor to RSA BSAFE, FIPS 140-2 validated cryptography library offered by RSA Security.

From 2004 to 2013 the default random number generator in the library contained an alleged backdoor from the NSA, as part of its secret Bullrun program.

Undisclosed until now was that RSA received $10 million in a deal that set the NSA formula as the preferred, or default, method for number generation in the BSafe software, according to two sources familiar with the contract. The backdoor was confirmed in the Snowden leaks in 2013.

NIST recommendations was also was affected.

We have no information on how this vulnerability can be used, which resources required and which libraries are affected. So, potentially, RSA libraries may still contain this backdoor.

To avoid this particular artificial vulnerability we have to avoid using of this random generator and check all libraries used (see list).

Weak keys generation with special random generator

Accordingly to this report, key generation is affected by random generator weakness and 2 of 1000 certificates (11 millions analyzed) allow attacker to recover secret key.

This is obviously result of RSA standard violation and sign that some crypto libraries are not following the standard due to various reasons (see previous section).

Backdoored keys generation with special seed for random generator

It's possible to generate key pair which will be "derived" from "secret key" and communication with use of such keys may be easily broken (private key may be restored without factorization problem) by the owner of this "secret key".

This is asymmetric backdoor designed to subvert RSA key generation algorithms by Ryan Castellucci. See also detailed explanation and sources, initial proof of concept and discussion on Reddit

The only way to detect signs of the problem is to detect an asymmetric Curve25519 backdoor in RSA key generation algorithms

Attack on Deffie-Hellman with pre-defined numbers

Accordingly to the report "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice" (13 Oct. 2015) "**We go on to consider Diffie-Hellman with 768- and 1024-bit groups. We estimate that even in the 1024-bit case, the computations are plausible given nation-state resources. A small number of fixed or standardized groups are used by millions of servers; performing precomputation for a single 1024-bit group would allow passive eavesdropping on 18% of popular HTTPS sites, and a second group would allow decryption of traffic to 66% of IPsec VPNs and 26% of SSH servers. A close reading of published NSA leaks shows that the agency's attacks on VPNs are consistent with having achieved such a break**."

By the first reports this vulnerability can be used by the government only with their new centers and such cluster will require few hundreds million dollars to be built.

Another opinion from Bruce Chneiers's blog "Another option is that the NSA has built dedicated hardware capable of factoring 1024-bit numbers. There's quite a lot of RSA-1024 out there, so that would be a fruitful project. So, maybe." Pay attention to the date, 3/22/2012

In 2015 other researches said that to do such computations attacker may use Amazon AWS and price for such attack will be 100k$, which is real.

Also attackers may use existing botnets for distributed computing or even systems like BOINC.

Two main problems made this scenario real:

- short key (1024)
- pre-defined parameters, used by many libraries to generate key pairs

We have no evidence that use of pre-defined numbers was the artificial vulnerability, but we can suggest this, if we will remember similar approaches for RSA (weakened random generator) and ECC (weakened curves).

Bottomline

As we can see, RSA algorithm is mathematically seems to be (see initial note) strong enough with proper padding, so most of known attacks are focused on the weak key generation.

Unfortunately, such attacks (backdoors) which are artificially added or discovered and not published by the government, may be potentially used by hackers.

So, security of the RSA algorithm currently **highly depends on the key generation process and libraries**. As we demonstrated, even certified libraries from RSA Security may contain backdoors or (as RSA Security rejected such imputations) weakness.

The only way to mitigate such security risks is to generate keys with trusted libraries or vendor (nevertheless, see note about RSA Security)

ECC Known Weakness

Mathematical weakness

Same is true for ECC, there is no practical attacks except side-channel attack which requires special conditions. Nevertheless, cryptographic experts expressed concerns that the National Security Agency had inserted a backdoor into at least one elliptic curve-based pseudo random generator.

Elliptic curve cryptography, as well as RSA, is vulnerable to quantum computing attacks, because vulnerable to modified Shor's algorithm for solving the discrete logarithm problem on elliptic curves. But currently it's a theoretical computation system, which is in research.

Implementation weakness

Situation is very similar as for RSA, artificial weaknesses and attacks focused on the key generation.

Recent articles in the media, based upon Snowden documents, have suggested that the NSA has actively tried to enable surveillance by embedding weaknesses in commercially-deployed technology -- including at least one NIST standard

Bruce Schneier has written that he has seen a bunch of secret Snowden documents, and after seeing them, he recommends classical integer discrete log-based cryptosystems over elliptic curve cryptography. When asked to elaborate on why he thinks we should avoid elliptic-curve cryptography, he writes: **“I no longer trust the constants. I believe the NSA has manipulated them through their relationships with industry.”**

Potentially weak curves from NIST

From this source: “This suggests we should look closely at how the "constants" (the curve parameters) have been chosen, if we use ECC. This is where things look concerning. I recently read a message on the tor-talk mailing list that seems to suggest the NIST curve parameters were not generated in a verifiable way. That message examines how the parameters were generated: *“I looked at the random seed values for the P-xxxr curves. For example, P-256r's seed is c49d360886e704936a6678e1139d26b7819f7e90. No justification is given for that value.”*

FIPS 186-3 may partially answer this particular concern in Appendix A and D.

So we have a lot of concerns and indirect evidences and negative opinions from crypto experts like Bruce Shneier, but there is no explicit evidences that NIST curves can’t be trusted.

The only known to me work which explains why NIST curves may be weak is well-know work from Daniel J. Bernstein, Tanja Lange “Security dangers of the NIST curves”.

Good starting point to dive into the problem - “Is there a feasible method by which NIST ECC curves over prime fields could be intentionally rigged?”

Brief summary of all suspicious is available in “Why I don't Trust NIST P-256” article.

We can’t estimate resources and prerequisites required to use this vulnerability, is it can be used by the government only or it can be used by hackers also.

Bottomline

As was said, we have no explicit evidences that NIST curves can't be trusted. But we have a lot of suspicious and opinions.

Also we can't estimate, are ECC vulnerabilities can be used by hackers.

The only way to mitigate this risk is to use trusted libraries to generate keypairs and choose trusted curves. Another approach is blindly trust NIST and NSA that such artificial vulnerabilities can be used by the government only.

Recommendations on Key length

The ECRYPT II recommendations on key length say that a 128-bit symmetric key provides the same strength of protection as a 3,248-bit asymmetric key. And that those key lengths provide long term protection of data encrypted with them. More recommendations here http://www.keylength.com/en/3/

This means, that our RSA key must be 3248 bits or more. For ECC we need no less than 256 bits. It's a minimum for "Long-term protection" until 2040, by ECRYPT II Recommendations (2012)

Recommended key lengths are:
256 bits for AES
15424 bits for RSA
512 bits for ECC

Performance

Basically, asymmetric encryption algorithms are significantly slower than symmetric encryption algorithms, so this is the reason why hybrid systems are used.

RSA is slower than ECC and ECC is slower than most of modern symmetric algorithms.

By the way, we have to keep in mind that for **hybrid schema** we need to encrypt key for symmetric algorithm, so if data size is comparable to symmetric algorithm key size, then hybrid schema will be slower than pure PKE.

Pure symmetric encryption is faster for all cases.

If data size is less than 256 bits (secure key for AES), then pure PKE will be faster

If data size is more than 256 bits, then hybrid schema will become faster after some data size.

Algorithms and message size

RSA

Basically, due to nature of RSA, output is a set of block with length same as key length plus padding to avoid **chosen ciphertext attack**, see OAEP.

So, output will be no less than key length. For 2048 bits key, every encrypted value will be no less than 256 bytes.

ECC

Like RSA, output is a set of block with length same as key length.

So, output will be no less than key length. For 256 bits key, every column will be no less than 32 bytes.

Hybrid systems with RSA

Hybrid systems, like PGP, will have similar issues because generated key for symmetric encryption will be encrypted with PKE, and data will be encrypted by symmetric algorithm.

For RSA with 2048 bit key + AES256, message will be 256 (RSA part) + n*32 (AES part) bytes (without HMAC) minimum. Then, in general, will expand linearly (by 32 bytes blocks).

This means that output for hybrid system with RSA2048 and AES256 is 256+32 bytes minimum, and will expand in 32-bytes blocks.

Hybrid systems with ECC

The same as for Hybrid systems with RSA is true for Hybrid systems with ECC.

For ECC with 256 bit key + AES256, message will be 32 (RSA part) + n*32 (AES part) bytes (without HMAC) minimum. Then, in general, will expand linearly (by 32 bytes blocks).

This means that output for hybrid system with ECC and 256-bit key and AES256 is 32+32 bytes minimum, and will expand in 32-bytes blocks.

Bottomline

ECC and RSA are recommended as PKE algorithms, where RSA is more mature, ECC is faster and generates significantly less output.

PGP with RSA or ECC and AES256 for symmetric encryption is recommended as Hybrid system.

On the data size less than 256 bits, pure PKE will be faster than Hybrid system.

To choose between RSA and ECC (or hybrid system based on RSA or ECC) we may need additional investigation “RSA vs ECC”, but, in general, both are described in RFC for PGP and widely used, both have no practical attacks described.

Key length 3248 bits or more should be used for RSA and 256 bits or more for ECC.

Recommended key lengths are:
256 bits for AES
15424 bits for RSA
512 bits for ECC