

BREAKING RSA & USING ASYMMETRIC CRYPTO

Luke Anderson

luke@lukeanderson.com.au

15th April 2016

University Of Sydney



1. Crypto-Bulletin

2. Breaking RSA

2.1 Chinese Remainder Theorem

2.2 Common Attacks

- Factoring Attack

- Small Encryption Exponent Attack

- Small Decryption Exponent & Forward Search Attack

2.3 Homomorphic Properties of RSA

2.4 Size of Modulus in RSA

2.5 Alternative: Rabin Cryptosystem

3. RSA Background and Methods

3.1 Useful Methods

- Square and Multiply

- Addition Chains

3.2 Complexity Theory

4. Symmetric vs. Asymmetric

4.1 Symmetric Crypto

4.2 Asymmetric Crypto

4.3 Combining Cryptosystems

4.4 Symmetric Key Sizes

4.5 Interesting breaks on Symmetric Crypto

4.6 Asymmetric Key Sizes

4.7 Considerations for Key Sizes

CRYPTO-BULLETIN

New Threat Can Auto-Brick Apple Devices

(< iOS 9.3.1)

<http://krebsonsecurity.com/2016/04/new-threat-can-auto-brick-apple-devices/>

Microsoft patches much-hyped Badlock bug

<http://www.itnews.com.au/news/microsoft-patches-much-hyped-badlock-bug-418121>

FBI: \$2.3 Billion Lost to CEO Email Scams

<http://krebsonsecurity.com/2016/04/fbi-2-3-billion-lost-to-ceo-email-scams/>

BREAKING RSA

Chinese Remainder Theorem (CRT)

The Chinese Remainder Theorem (CRT) can be used to break RSA.

Without knowing the factorisation of $n = pq$, you can use CRT to solve a system of equations:

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

$$x \equiv a_3 \pmod{m_3}$$

Where $\gcd(m_i, m_j) = 1$
(i.e. m_i & m_j are relatively prime).

The CRT states that there exists a simultaneous solution to these equations where any two such solutions are congruent to each other mod N , where $N = n_1 \cdot n_2 \cdot n_3 \cdots$

Chinese Remainder Theorem (CRT)

For each i , define:

$$M_i = \frac{M}{m_i} = \prod_{i \neq j} m_j$$

By the Euclidean algorithm, calculate N_i such that:

$$N_i M_i = 1 \bmod m_i$$

The solution to the system of simultaneous equations is:

$$x = \sum_{i=1}^r a_i M_i N_i$$

Chinese Remainder Theorem (CRT): Example

Solve the following system of equations:

$$x \equiv a_1 \pmod{7}$$

$$x \equiv a_2 \pmod{11}$$

$$x \equiv a_3 \pmod{13}$$

Chinese Remainder Theorem (CRT): Example

Now $M = m_1 \cdot m_2 \cdot m_3 = 7 \times 11 \times 13 = 1001$

To find N s (e.g. for $i = 1$, N_1):

$$N_i M_i = 1 \bmod m_i \text{ where: } M_1 = 11 \times 13$$

$$N_1 \times (11 \times 13) = 1 \bmod 7 \text{ where: } 11 \times 13 = 3 \bmod 7$$

$$N_1 \times 3 = 1 \bmod 7$$

Using Euler's generalisation: $m_{\phi(n)} \equiv 1 \bmod n$:

$$\Phi(m_1) = 7 - 1 = 6$$

So:

$$N_1 = 3^{\Phi-1} \bmod 7$$

$$= 3^5 \bmod 7$$

$$= 5 \bmod 7$$

Chinese Remainder Theorem (CRT): Example

Solving for other N s:

$$M_1 = 143 \qquad N_1 = 5 \qquad (11 \times 13)$$

$$M_2 = 91 \qquad N_1 = 4 \qquad (7 \times 13)$$

$$M_3 = 77 \qquad N_3 = 12 \qquad (11 \times 7)$$

So:

$$x = \sum_{i=1}^r a_i M_i N_i$$

$$x = (715a_1 + 364a_2 + 924a_3) \bmod 1001$$

Chinese Remainder Theorem (CRT): Example

Lets say Alice sends the same message (p) to three recipients, whose public keys are:

$$(n_1, e), (n_2, e), (n_3, e)$$

e.g. $p = 10$ and $e = 3$

Public keys are: $(87, 3), (115, 3), (187, 3)$

Then:

$$c_1 = p^e \bmod n_1 = 10^3 \bmod 87 = 43$$

$$c_2 = p^e \bmod n_2 = 10^3 \bmod 115 = 80$$

$$c_3 = p^e \bmod n_3 = 10^3 \bmod 187 = 65$$

Chinese Remainder Theorem (CRT): Example

By the previous method:

$$M_1 = 115 \times 187 = 21505$$

$$N_1 = 49$$

$$M_2 = 87 \times 187 = 16269$$

$$N_2 = 49$$

$$M_3 = 87 \times 115 = 10005$$

$$N_3 = 2$$

and: $M = 87 \times 115 \times 187 = 1870935$

Then: $p^e \bmod m_i = c_i \bmod m_i$

So: $x = c_i \bmod m_i$

$$\begin{aligned}x &= \sum c_i M_i N_i \\&= (43 \times 21505 \times 49) + (80 \times 16269 \times 49) + (65 \times 10005 \times 2) \\&= 110386165 \\&= 1000 \bmod 1870935 \\&= m^3 \bmod M\end{aligned}$$

$$m = \sqrt[3]{x}$$

Common Attacks on RSA

Some common attacks on RSA:

- Factoring Attack
- Small exponent attacks:
 - Small encryption exponent
 - Small decryption exponent
- Forward search attack
- Homomorphic attack
- Common modulus attack
- Cycling attack
- Message concealing (sec. 8.2.2 viii)
- Small size of the modulus
- Bad selection of primes

Factoring Attack on RSA

The RSA problem is recovering m from $c = m^e \bmod n$ with only n and e .

Suppose n can be factored into p and q : Then $\Phi(n) = (p-1)(q-1)$ can be computed – the Euler's totient function.

Therefore d can be computed as: $ed \equiv 1 \bmod \Phi(n)$

Therefore we can recover the message m .

FACT

The problem of computing the RSA decryption exponent from the public key (n, e) , and the problem of factoring n are computationally equivalent.

When performing key generation, it is imperative that the primes p and q are selected to make factoring $n = pq$ very difficult
e.g. by picking a p and q that are roughly equal size

Small Encryption Exponent Attack (RSA)

The small encryption exponent attack, or [Coppersmith's attack](#), exploits a bad choice of encryption exponent.

A small encryption exponent is often chosen in order to improve the speed of RSA encryption.

e.g. $2^{16} + 1$ is often used

If a group of entities all use the same encryption exponent, it is clear that they must have their own distinct modulus. Otherwise, if they were the same modulus, users could obviously easily calculate the other users' private keys (d).

Say Alice wishes to send messages to three parties, all with a small encryption exponent: $e = 3$

$$c_1 = m^3 \bmod n_1$$

$$c_2 = m^3 \bmod n_2$$

$$c_3 = m^3 \bmod n_3$$

Small Encryption Exponent Attack (RSA)

Observing c_1, c_2, c_3 and knowing n_1, n_2, n_3 , we use the CRT:

$$x = m^3 \bmod (n_1 \cdot n_2 \cdot n_3)$$

Since $m < n_i$ for $\forall n$ (otherwise information would be lost in encryption)

$$x = m^3$$

$$m = x^{\frac{1}{3}} = \sqrt[3]{x}$$

Thus, a small encryption exponent should not be used to send the same message (or the same message with variation) to several entities.

NOTE

Salting the plaintext (padding with random bits) can help avoid this attack.

Small Decryption Exponent (**Wiener's Attack**)

- A small decryption exponent should also be avoided ($d < \frac{N^{\frac{1}{4}}}{3}$).
- Common mistake in small devices, since a small d makes for efficient decryption.

Forward Search Attack

- Since the encryption key is public, if the message space is small or predictable, an attacker can try to brute force on the message space. Salting the plaintext may help prevent this attack
- Example: A stock trading system
 - Message format: ``{BUY, SELL} DDDD TICKER'' (only 1000 tickers)
 - $|M| = 2 \times 10000 \times 1000 = 20,000,000$ possible messages
 - At 1 million guesses / second \rightarrow 20 seconds to guess transmitted message.

Homomorphic Properties of RSA

RSA encryption is **homomorphic**.

Suppose:

$$c_1 = m_1^e \bmod n$$

$$c_2 = m_2^e \bmod n$$

Then:

$$c_1 \times c_2 = (m_1 \times m_2)^e \bmod n$$

Using this property, we can attack RSA.

Homomorphic Properties of RSA

Suppose we want Alice to reveal the decryption of:

$$c = m^e \bmod n$$

Bob sends Alice:

$$(c') = cx^e \bmod n$$

Where x is a randomly chosen “blinding factor”.

Alice computes:

$$\begin{aligned}(c')^d &= (cx^e)d \bmod n \\ &= c^d x^{ed} \bmod n \\ &= mx^{ed} \bmod n \\ &= mx \bmod n\end{aligned}$$

If Alice reveals this information, Bob can “unblind” the message:

$$m = (mx)x^{-1} \bmod n$$

Size of Modulus in RSA

Powerful attacks on RSA include using a [quadratic sieve](#) and [number field sieve](#) factoring algorithms to factor the modulus $n = pq$.

- 1977:** a 129 digit (426 bit) RSA puzzle was published by Martin Gardner's Mathematical Games in Scientific American. Ron Rivest said RSA-125 would take "40 quadrillion years". In 1993 it was cracked by a team using 1600 computers over 6 months: "[the magic words are squeamish ossifrage](#)".
- 1999:** a team led by de Riele [factored a 512 bit number](#)
- 2001:** Dan Bernstein wrote a paper proposing a circuit-based machine with active processing units (the same density as RAM) that could factor keys roughly 3 times as long with the same computational cost – so is 1536 bits insecure??
 - The premise is that algorithms exist where if you increase the number of processors by n , you decrease the running time by a factor greater than n .
 - Exploits massive parallelism of small circuit level processing units
- 2009:** [RSA-768](#), 232 digits (768 bits), is factored over a span of 2 years.

Size of Modulus in RSA

In **2012**, a 1061 bit (320 digit) special number ($2^{1039} - 1$) was factored over 8 months (a special case), using a “special number field sieve”.

- Unthinkable in 1990
- We have:
 - Developed more powerful computers
 - Come up with better ways to map the algorithm onto the architecture
 - Taken better advantage of cache behaviour
- “Is the writing on the wall for 1024-bit encryption?
The answer to that is an unqualified yes” – [Lenstra](#)

It is recommended today that 4096 bit keys be used (or at least 2048 bit) and p and q should be about the same bit length (but not *too* close to each other).

Advances in factoring are leaps and bounds over advances in brute force of classical cyphers:

http://en.wikipedia.org/wiki/RSA_Factoring_Challenge

Alternative: Rabin Cryptosystem

The [Rabin cryptosystem](#) is similar to RSA, but has been proven to be as hard as the *integer factorization* problem.

Public key: $n = pq$

Private key: p, q (roughly the same size)

Encryption: $c = m^2 \bmod n$

Decryption:

Calculate the four square roots: m_1, m_2, m_3, m_4 of c

The message sent was one of these roots.

The security is based on the fact that finding square roots mod n without knowing the prime factorisation of n is computationally equivalent to factoring.

RSA BACKGROUND AND METHODS

Square and Multiply

In RSA and Discrete Log, a common operation is exponentiation.
i.e. calculating g^e where g and e are large numbers (≈ 300 digits)

A simple approach to this is to use **square-and-multiply**:

$$g^{23} = g^{16} \cdot g^4 \cdot g^2 \cdot g^1$$

In this example, we use 7 multiplications (assuming squaring is computationally equivalent to multiplying).

In Python:

Addition Chains

Using [addition chains](#), we can be a little more efficient:

$$g^1 \cdot g^2 \cdot g^3 \cdot g^5 \cdot g^{10} \cdot g^{20} \cdot g^{23}$$
$$g^2 \cdot g^3 = g^5 \text{ etc.}$$

This take only 6 multiplications (versus 7 for square-and-multiply).
An addition chain is used to minimise the number of multiplications required.

The addition chain of length s for exponent e is a sequence of positive integers $\{u_0, \dots, u_s\}$ and associated sequence $\{w_0, \dots, w_s\}$ of pairs of integers $w_i = (i_1, i_2)$ with the property that:

$$u_0 = 1, u_s = e$$

$$u_i = u_{i_1} + u_{i_2}$$

Addition Chains: Example

Take $e = 15$ (i.e. calculate g^{15}).

The addition chain can be represented as:

$$g^{15} = g \times (g \times [g \times g^2]^2)^2 \quad \text{binary - 6 multiplications}$$

$$g^{15} = g^3 \times ([g^3]^2)^2 \quad \text{shortest - 5 multiplications}$$

Finding the shortest chain is computationally hard though ([NP-hard](#))

It is akin to solving the [travelling salesman problem](#).

Complexity Theory Overview

Fact: $P \subseteq NP$ ¹

Unknown: Is $P = NP$?

Example **NP** problem:

- Given a positive integer n , is n composite?
i.e. are there integers $a, b > 1$ such that $n = ab$?

If L_1 and L_2 are two decision problems, L_1 is said to **polynomial reduce** to L_2 ($L_1 \leq^P L_2$) if there is an algorithm that solves L_1 , which uses an algorithm that solves L_2 as a subroutine, and runs in polynomial time.

Two problems are said to be computationally equivalent if:

$$L_1 \leq^P L_2 \text{ and } L_2 \leq^P L_1$$

¹**P** versus **NP**

SYMMETRIC vs. ASYMMETRIC

Summary: Symmetric Crypto

Advantages of symmetric-key crypto:

- Can be designed to have high throughput rates
- Keys are relatively short
(128, ..., 256 bits)
- Can be used as primitives to create other constructs such as pseudo random number generators (PRNGs).
- Can be used to construct stronger cyphers
e.g. simple substitutions and permutations can be used to create stronger cyphers
- All known attacks involve “exhaustive” key search.

Disadvantages of symmetric-key crypto:

- In a two party network, the key must remain secret at both ends
- Sound practice dictates the key needs to be changed frequently
(e.g. each session)
- In a large network, $n!$ keys are required, which creates a massive problem for key management.

Summary: Asymmetric Crypto

Advantages of asymmetric-key crypto:

- Only the private key needs to remain secret
- The administration of keys on a network requires the presence of only one functionally trusted (honest and fair) TTP².
- Depending on the mode of usage, the public and private key pairs may be used for long periods of time (upper bound: Moore's Law)
- In large networks, n keys are required instead of $n!$

Disadvantages of asymmetric-key crypto:

- Throughput rates are typically very slow (for all known algorithms)
- Key sizes are typically much larger (1024, ..., 4096 bits)
- Security is based upon the *presumed* difficulty of a small set of number-theoretic problems; all known are subject to short-cut attacks (e.g. knowing the prime factorisation of n)
- Public key crypto does not have as much of an extensive history in the public world, compared to symmetric.

²Trusted Third Party

Combining Cryptosystems

Symmetric and asymmetric crypto are complementary.

Asymmetric crypto can be used to establish a key for subsequent faster symmetric crypto
(e.g. a session key)

Alice and Bob can take advantage of the long term benefits of public key crypto by publishing their public keys to a directory.

Public key crypto is good for key management and signatures.

Private key crypto is good for encryption and some data integrity applications.

Symmetric Crypto: Key Length

Security of a symmetric cypher is based on:

- strength of the algorithm
- length of the key

Assuming the strength of the algorithm is perfect (impossible in practice) then brute force is the best attack.

Cost (USD)	40 bits	56 bits	64 bits	128 bits
\$100k	0.06 sec	1.1 hrs	11.5 days	10^{18} years
\$1M	6.25 ms	6.5 mins	1.2 days	10^{17} years
\$100M	0.06 ms	3.75 mins	17 mins	10^{15} years
\$1B	$6.25 \mu s$	0.4 sec	1.9 mins	10^{14} years

Table: Hardware Attack Estimates (2005)

Interesting ways to break symmetric ciphers

Virus / Worm:

- What if a bot net forced a cypher?
- Melissa infected $\approx 800k$ computers
- If we can cracking DES @ 280k keys/s (P4 @ 2.8 Ghz)
- Mlissa-DES could brute force the key in 30 hours
- In 2000, a worm did just that.
- Newer botnets have many more computers.

Chinese Lottery

- Say a 1M key/s chip was built into every radio and TV in China
- Each chip is designed to brute force a key sent over the air
- If 10% of the people in China have a radio or TV:
the 56 bit DES key space can be exhausted in 12 minutes

In future lectures, we'll see how Bitcoin extends on these concepts.

Asymmetric Crypto Key Length

Security of all known public key algorithms is based on the presumed difficulty of a small set of number-theoretic problems.

All are subject to short cut attacks. Tomorrow we might find a way to factor easily.

e.g. Quantum? DNA?

1977: Ron Rivest (RSA) said factoring 125 digit number would take 4×10^{16} years

2003: 576 bit (177 digit) number factored

2012: 1061 bit (320 digit) number factored

Designs are being drawn out for optical / quantum sieving machines that could lead to massive optimisations on these numbers in the near future.

How long should an asymmetric key be?

Protection	Symmetric	Asymmetric	Hash
Attacks in real time by individuals	32	-	-
Short term protection against small organisations	64	816	128
Short term protection against medium organisations	72	1008	144
Very short term protection against agencies, long term protection against small organisations (2016 – 2017)	80	1,248	160
Legacy standard level (2016 – 2020)	96	1,776	192
Medium-term protection (2016 – 2030)	112	2,432	224
Long-term protection (2016 – 2040)	128	3,248	256
“Foreseeable future” good protection against quantum computers unless Shor’s algorithm applies	256	15,424	512

Minimal key sizes for different types of crypto (all sizes are in bits)

Considerations for Key Sizes

Type of Traffic	Lifetime	Min. Key Length
Tactical Military Information	Minutes / hours	64 bits
Product Announcements or M&A	Days / weeks	80 bits
Long Term Business Plans	Years	96 bits
Trade Secrets (e.g. Coca Cola)	Decades	128 bits
H-bomb Secrets	> 40 years	128 – 192 bits
Identities of Spies	> 50 years	128 – 192 bits
Personal Affairs	> 50 years	128 – 192 bits
Diplomatic Embarrassments	> 65 years	192 bits
U.S. Census Data	100 years	256 bits

All key sizes are optimistic. Five or ten years ago people would've suggested smaller key sizes for these tasks.