The EJ cryptosystem

Efraem Joji
efraemjoji@gmail.com

The EJ (Efraem Joji's) cryptosystem is a public-key cryptosystem with key exchange, asymmetric encryption and digital signature. It is based on the associative binary relation of square matrix multiplication, with all matrix elements reduced modulo a prime number (here, chosen to be the mersenne prime $p=2^{31}-1$).

It currently gives about 256 bits security for a 288 bit key , that is, double the bits of security of Elliptic Curve Cryptography and about 14 times that of RSA(Rivest-Shamir-Adleman) . It is about 24 times faster than the RSA and about 9 times faster than ECDH (Elliptic Curve Diffie Hellman) .

Theory

The group of n x n matrices with all elements reduced modulo a prime (to make it a finite group and make it feasible for computation) and with the operation of matrix multiplication (which is an associative operation) is a cyclic group, and we can use the multiplicative notation for its elements.[1]

Therefore, the Diffie-Hellman key exchange will work in such a group [2]. Further, due to the multiplicative notation usable here, a schnorr-like signature algorithm also works. This will be proved in later parts of this text, using the multiplicative notation.

The following notation will be used in this text:
B = the base matrix, a universally known square matrix
p = a prime number, modulo which all elements of the matrices will be reduced
* = the binary relation of matrix multiplication followed by reduction modulo p of all the elements of the resulting matrix.

M^n refers to $M*M*...*M$ ("n" times) ie. multiplying the matrix "M" with itself "n" times, which can be computed in $O(\log n)$ time with binary exponentiation.

H(m) refers to the cryptographically secure hash function H applied to a string "m"

$E(y,m)$ refers to the secure symmetric encryption function E applied to a plaintext message m , with y as the key, in order to produce the ciphertext output.
$D(y,n)$ refers to the corresponding secure symmetric decryption function D applied to a ciphertext message n , with y as the key, in order to produce the plaintext output.
|| refers to concatenation of strings

Key generation

A securely and randomly generated number x is chosen as the private key.

$P=B^x$ is computed and is taken as the public key

Key exchange algorithm

This is based on a generalization of the Diffie-Hellman key exchange.

1) Alice and Bob randomly and securely generate j and k , their respective public keys.
2) They compute $A_1=B^j$ and $A_2=B^k$, their respective private keys and exchange them through an insecure channel.
3) Alice computes $S_1=(A_2)^j=(B^k)^j=B^{kj}$ and Bob computes $S_2=(A_1)^k=(Bj)k=Bjk$, their secret keys.

As $S_1=S_2$ (Using the multiplicative notation), the key exchange is successful and they have established a shared secret.

Asymmetric encryption algorithm

Let Bob's public key be $Q=B^x$, where x is his private key .

1) Alice wishes to send Bob a message. She computes a nonce k, secret $S_1=Q^k$ and $R=B^k$
2) Then, she computes the ciphertext message $c=E(S_1,m)$ and sends (R, c) to Bob.
3) Bob computes $S_2=R^x$ and decrypts $m=D(S1,c)$.

Proof: $S_2=R^x=(B^k)^x=B^{kx}$ and $S_1=Q^k=(B^x)^k=B^{xk}$ are equal.

Digital signature

Let Bob's public key be $P=B^x$ where x is his private key. To sign a message m, preferably being a secure cryptographic hash of the original message :

1) Bob randomly and securely generates a non-reusable nonce k and computes $Q=B^k$ $e=H(Q||m||Q)$ (or some other secure function tightly mixing m and Q with H)
2) He computes $r=k-xe$ and sends the pair (r,e) as his signature, along with m .

To verify the signature,

1) Compute $Q_1=B^r*P^e$.
2) Compute $e_1=H(Q_1||m||Q_1)$. Iff $e_1=e$, the signature is verified, otherwise it is rejected.

Proof:

$$Q_1=B^r*P^e=B^{(k-xe)}*(B^x)^e=(\underbrace{B*B*...*B}_{k-xe\ \ times})*(\underbrace{B*B*...*B}_{xe\ \ times})=(\underbrace{B*B*...*B}_{\text{k times}})=B^k=Q$$

So, $e_1=H(Q_1||m||Q_1)=e=H(Q||m||Q)$ if and only if the signature is valid ie. iff $Q=Q_1$.

Security

The basic assumption underlying the cryptosystem is that it is hard to find j, k or B^{jk} , given B , $A_1=B^j$ and $A_2=B^k$, which means solving the discrete logarithm or Diffie-Hellman problem in the group of $n\,x\,n$ matrices with all elements reduced modulo a prime (p) and with the operation of matrix multiplication .

This is almost as hard as brute force, the best currently known time complexity for such an algorithm is $O(2^n/p)$.

This is because $|B^x| mod\ p = |B^x| mod\ p$ (a property of determinants) , the value (x mod p) can be computed relatively easily. But, the value of x remains hidden, and the search becomes just marginally easier than brute force , as $\log(x) \gg \log(k)$ in all use cases.

A typical implementation ie. the one that is now programmed in C++ , accessible at https://github.com/EfraemJoji/EJcryptosystem has $0<x<2^{288}$ and $p=2^{32}-1$ ie. the bits of security offered is over $288-32=256$ bits of security for a 288 bit private key and 288 bit public key . This is much better compared to most existing cryptosystems. (eg. Elliptic Curve Cryptography with a 256 bit key and RSA with a 3072 bit key give just 128 bits of security [3] [4])

Further, the EJ cryptosystem is likely to resist quantum attacks (while most Elliptic Curve Cryptography and the RSA have known quantum computing attacks), and the EJ cryptosystem works in a non-commutative group, which may increase security.

Speed

→ RSA 4096 bits is almost comparable in bits of security to EJ cryptosystem 288 bits

```
$ openssl speed rsa4096
sign/s = 59.6   verify/s = 3825.3
```

→ ECDH - 521 bit is almost comparable in bits of security to EJ cryptosystem 288 bits

```
$ openssl speed ecdh
Doing 521 bits  ecdh's for 10s: 10994 521-bits ECDH ops in 10.00s
```

→ EJ cryptosystem 288 bits (not assembly optimized, but still considerably faster)

```
$ g++ -O2 -m64 test.cpp ; ./a.out
100368 operations per second.
```

Test machine:
OS: Debian GNU/Linux 10 (buster)
CPU: Intel Pentium N4200 Quad Core , 64 bits L2 cache: 1024 KiB
flags: lm nx pae sse sse2 sse3 sse4_1 sse4_2 ssse3 vmx
Speed: 0.8 – 2.5 GHz

Model: Acer Aspire ES1-533

References

[1] https://en.wikipedia.org/wiki/Cyclic_group

[2] Buchmann, Johannes A. (2013). Introduction to Cryptography (Second ed.). Springer Science+Business Media. pp. 190–191. ISBN 978-1-4419-9003-7.

[3] Barker, Elaine (2016-01-28). "NIST Special Publication 800-57 Part 1 Revision 4: Recommendation for Key Management: General" (PDF). National Institute of Standards and Technology: 53. doi:10.6028/NIST.SP.800-57pt1r4 - https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[4] Barker, Elaine; Dang, Quynh (2015-01-22). "NIST Special Publication 800-57 Part 3 Revision 1: Recommendation for Key Management: Application-Specific Key Management Guidance" (PDF). National Institute of Standards and Technology: 12. doi:10.6028/NIST.SP.800-57pt3r1. Retrieved 2017-11-24. - https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf