Python and Crypto: Ring Signatures

Prof Bill Buchanan OBE, The Cyber Academy

http://asecuritysite.com
I know one of you leaked the information. But which of you was it?

We have a secret.
Signed,
A member of the crypto team
Generating Signature

Bob generates random numbers for Trent, Alice and Eve ($s_1, s_3$ and $s_4$).
Generating Signature

Original seed generated: \( u \)

\[
E_k = \text{Hash}(\text{Message})
\]

\[
v = E_k(u)
\]

\[
v = E_k(s_1 P_1 \oplus v)
\]

Fake secret key for Trent \( (s_1) \)

\[
v = E_k(s_3 P_3 \oplus v)
\]

Fake secret key for Eve \( (s_3) \)

\[
v = E_k(s_4 P_4 \oplus v)
\]

Fake secret key for Alice \( (s_4) \)

Bob now adds:

\[
v = E_k(u \oplus v)^d
\]

All the operations are conducted with \( \text{mod } N_i \)

Bob sends: \{Message, v, P_1, P_2, \ldots P_r, s_1, s_2, \ldots s_i\}
Verifying The Signature

Original seed generated: u

\[ E_k = \text{Hash(Message)} \]
\[ v = E_k(u) \]
\[ v = E_k(s_1 \oplus v) \]  
Fake secret key for Trent (s_1)
\[ v = E_k(s_3 \oplus v) \]  
Fake secret key for Trent (s_3)
\[ v = E_k(s_4 \oplus v) \]  
Fake secret key for Trent (s_4)

Bob now adds:

\[ v = E_k(u \oplus v)^d \]  
All the operations are conducted with \((\mod N_i)\)

\[ v = E_k((u \oplus v)^{\chi_a} \rightarrow u) \]
Verifying The Signature

Original seed generated: $u$

$E_k = \text{Hash(Message)}$

$\nu = E_k(u)$

$v = E_k(s_i^{p_1} \oplus \nu)$ \hspace{1cm} \text{Fake secret key for} \hspace{1cm} \text{Trent} (s_i)$

$v = E_k(s_i^{p_2} \oplus \nu)$ \hspace{1cm} \text{Fake secret key for} \hspace{1cm} \text{Trent} (s_i)$

$v = E_k(s_i^{p_4} \oplus \nu)$ \hspace{1cm} \text{Fake secret key for} \hspace{1cm} \text{Trent} (s_i)$

Bob now adds:

$v = E_k(u \oplus \nu)^d$ \hspace{1cm} \text{All the operations are} \hspace{1cm} \text{conducted with} \hspace{1cm} \text{(mod $N_i$)}$

In the code:

```python
def sign(self, m, z):
    self.permut(m)
    s = [None] * self.n
    u = random.randint(0, self.q)
    c = v = self.E(u)
    for i in (range(z+1, self.n) + range(z)):
        s[i] = random.randint(0, self.q)
        e = self.g(s[i], self.k[i].e, self.k[i].n)
        v = self.E(v^e)
        if (i+1) % self.n == 0:
            c = v
        s[z] = self.g(v^u, self.k[z].d, self.k[z].n)
    return [c] + s

def verify(self, m, X):
    self.permut(m)
    def _f(i):
        return self.g(X[i+1], self.k[i].e, self.k[i].n)
    y = map(_f, range(len(X)-1))
    def _g(x, i):
        return self.E(x^y[i])
    r = reduce(_g, range(self.n), X[0])
    return r == X[0]
```
def sign(self, m, z):
    self.permut(m)
    s = [None] * self.n
    u = random.randint(0, self.q)
    c = v = self.E(u)
    for i in (range(z+1, self.n) + range(self.n)):
        s[i] = random.randint(0, self.q)
        e = self.g(s[i], self.k[i].e, self.k[i].n)
        v = self.E(v^e)
        if (i+1) % self.n == 0:
            c = v
    s[z] = self.g(v^u, self.k[z].d, self.k[z].n)
    return [c] + s

def verify(self, m, X):
    self.permut(m)
    def _f(i):
        return self.g(X[i+1], self.k[i].e, self.k[i].n)
    y = map(_f, range(len(X)-1))
    def _g(x, i):
        return self.E(x^y[i])
    r = reduce(_g, range(self.n), X[0])
    return r == X[0]
So what?

The major problem with the Bitcoin network, is that the amount of a transaction and the sender and receive of the funds are not private, and someone who knows someone's address can trace their transactions. This is the case because the blockchain needs to check that the sender has enough funds to pay the recipient. Thus many cryptocurrencies are looking for ways of anonymising the transaction.
So what?

The major problem with the Bitcoin network, is that the amount of a transaction and the sender and receive of the funds are not private, and someone who knows someone's address can trace their transactions. This is the case because the blockchain needs to check that the sender has enough funds to pay the recipient. Thus many cryptocurrencies are looking for ways of anonymising the transaction.
The method proposed by Rivest et al uses RSA is not efficient for modern systems.

Thus Greg Maxwell’s defined an elliptic curve methods as a new way of creating the ring signature: the Borromean ring signature [paper].

The cryptocurrency Monero adopted the method for anonymising transactions.

Since migrated to a new method: Multi-layered Linkable Spontaneous Anonymous Group signature. This method hides the transaction amount and the identity of the payer and recipient [paper]. It is now known as RingCT (Ring Confidential Transactions), and was rolled-out in January 2017 and mandatory for all transactions from September 2017.
Ring Signatures

I know one of you leaked the information. But which of you was it?

We have a secret.

Signed,
A member of the crypto team
Python and Crypto: Ring Signatures

Prof Bill Buchanan OBE, The Cyber Academy

http://asecuritysite.com