Gröbner Bases In Public Key Cryptography: Hope Never Dies

M. Caboara, F.Caruso, **C. Traverso**

Eurocrypt 2008 Rump session
İstanbul, April 15, 2008

The Challenge

Boo Barkee, Deh Cac Can, Julia Ecks, Theo Moriarty, R. F. Ree:

Why you cannot even hope to use Groebner Bases in Public Key Cryptography: an open letter to a scientist who failed and a challenge to those who have not yet failed[1],

Journal of Symbolic Computation, 18 (6) 1994

[1] partially supported by Spectre

The Challenge

Boo Barkee, Deh Cac Can, Julia Ecks, Theo Moriarty, R. F. Ree:

Why you cannot even hope to use Groebner Bases in Public Key Cryptography: an open letter to a scientist who failed and a challenge to those who have not yet failed[1],

Journal of Symbolic Computation, 18 (6) 1994

In the 14 years since the publication of this paper, several scientists have failed while trying to counter this criminal threat, including eminent cryptographers like M.R. Fellows, N. Koblitz, (*Combinatorial Cryptosystems Galore!*) and their epigones that defined several Polly Cracker cryptosystems. None survived.

[1] partially supported by Spectre

The Challenge

Boo Barkee, Deh Cac Can, Julia Ecks, Theo Moriarty, R. F. Ree:

Why you cannot even hope to use Groebner Bases in Public Key Cryptography: an open letter to a scientist who failed and a challenge to those who have not yet failed[1],

Journal of Symbolic Computation, 18 (6) 1994

In the 14 years since the publication of this paper, several scientists have failed while trying to counter this criminal threat, including eminent cryptographers like M.R. Fellows, N. Koblitz, (*Combinatorial Cryptosystems Galore!*) and their epigones that defined several Polly Cracker cryptosystems. None survived.

It is now our turn to risk to fail, proposing two new PK cryptosystems using Gröbner bases for the key definition.

[1] partially supported by Spectre

Two GB PK cryptosystems

Two GB PK cryptosystems:

- The two cryptosystems combine multivariate polynomial algebra and lattices, modifying two well-known cryptosystems:

Two GB PK cryptosystems

Two GB PK cryptosystems:

- The two cryptosystems combine multivariate polynomial algebra and lattices, modifying two well-known cryptosystems:
  - GGH by O. Goldreich, S. Goldwasser, and S. Halevi,

Two GB PK cryptosystems

Two GB PK cryptosystems:

- The two cryptosystems combine multivariate polynomial algebra and lattices, modifying two well-known cryptosystems:
  - GGH by O. Goldreich, S. Goldwasser, and S. Halevi,
  - NTRU by J. Hoffstein, J. Pipher, and J. H. Silverman.

Two GB PK cryptosystems:

- The two cryptosystems combine multivariate polynomial algebra and lattices, modifying two well-known cryptosystems:
  - GGH by O. Goldreich, S. Goldwasser, and S. Halevi,
  - NTRU by J. Hoffstein, J. Pipher, and J. H. Silverman.
- Both modifications change the key creation and decryption engine, but from the point of view of encryption they are the same as the original cryptosystems.

GB-GGH aka Lattice Polly Cracker

- The first cryptosystem modifies GGH, using the computation of the normal form with respect of a Gröbner basis (instead of Babai round-off algorithm) to decypher.

GB-GGH aka Lattice Polly Cracker

- The first cryptosystem modifies GGH, using the computation of the normal form with respect of a Gröbner basis (instead of Babai round-off algorithm) to decypher.
  Key ingredient: the equivalence of lattices and binomial ideals; $X^\alpha - X^\beta$ corresponds to the vector $\alpha - \beta$.

GB-GGH aka Lattice Polly Cracker

- The first cryptosystem modifies GGH, using the computation of the normal form with respect of a Gröbner basis (instead of Babai round-off algorithm) to decypher.
  Key ingredient: the equivalence of lattices and binomial ideals; $X^\alpha - X^\beta$ corresponds to the vector $\alpha - \beta$.
  The construction is complex, and very technical to ensure (conjectured) security, hence we cannot discuss it now.

GB-GGH aka Lattice Polly Cracker

- The first cryptosystem modifies GGH, using the computation of the normal form with respect of a Gröbner basis (instead of Babai round-off algorithm) to decypher.
  Key ingredient: the equivalence of lattices and binomial ideals; $X^\alpha - X^\beta$ corresponds to the vector $\alpha - \beta$.
  The construction is complex, and very technical to ensure (conjectured) security, hence we cannot discuss it now.
- The resulting cryptosystem is not only a lattice cryptosystem, but also a Polly Cracker cryptosystem; it resists all the known attacks, including the differential message attack of D. Hofheinz and R. Steinwandt that breaks all the other Polly Cracker cryptosystems.

GB-GGH aka Lattice Polly Cracker

- The first cryptosystem modifies GGH, using the computation of the normal form with respect of a Gröbner basis (instead of Babai round-off algorithm) to decypher.
  Key ingredient: the equivalence of lattices and binomial ideals; $X^\alpha - X^\beta$ corresponds to the vector $\alpha - \beta$.
  The construction is complex, and very technical to ensure (conjectured) security, hence we cannot discuss it now.
- The resulting cryptosystem is not only a lattice cryptosystem, but also a Polly Cracker cryptosystem; it resists all the known attacks, including the differential message attack of D. Hofheinz and R. Steinwandt that breaks all the other Polly Cracker cryptosystems.
- The remaining issue is the protection of the private key. We have tried several techniques, and discovered new attacks; we believe to have now a secure variant, but it has not yet undergone sufficient scrutiny.

NTRU

Concerning NTRU, we will give a few more details of our modification, that we called GB-NTRU.
This is an outline of NTRU:

- The public setting is given by n, q, p; $A = \mathbf{Z}^n/(x^n - 1)$ and the public computations are done in A/q.
- The private key is composed finding two “small” polynomials f, g and the public key is $h = p \cdot f_q^{-1} g \in A/q$
- The encyphering of a message m is $c = hr + m$, r random.
- The decyphering is made computing $fc \in A/q$, lifting to A, obtaining (if everything goes well) $fm + p \cdot hr = fm \in A/p$. Then $m \mod p$ is recovered.

In GB-NTRU we use bivariate (or multivariate) polynomials (this is needed for some technical constraints that will not be apparent in our talk).

These are the main differences in key creation:

GB-NTRU

These are the main differences in key creation:

- NTRU uses $A = \mathbf{Z}[x]/(x^n - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_q^{-1} g \in A/q$; q, p are public.

GB-NTRU

These are the main differences in key creation:

- NTRU uses $A = \mathbf{Z}[x]/(x^n - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_q^{-1} g \in A/q$; q, p are public.
- GB-NTRU uses $A = \mathbf{Z}[X]/(X^N - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_Q^{-1} g \in A/q$; q, p are public.

GB-NTRU

These are the main differences in key creation:

- NTRU uses $A = \mathbf{Z}[x]/(x^n - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_q^{-1} g \in A/q$; q, p are public.
- GB-NTRU uses $A = \mathbf{Z}[X]/(X^N - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_Q^{-1} g \in A/q$; q, p are public.
  $q \in Q \subseteq A$

GB-NTRU

These are the main differences in key creation:

- NTRU uses $A = \mathbf{Z}[x]/(x^n - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_q^{-1} g \in A/q$; q, p are public.
- GB-NTRU uses $A = \mathbf{Z}[X]/(X^N - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_Q^{-1} g \in A/q$; q, p are public.

  $q \in Q \subseteq A$

  X is a pair of variables (x, y), Q is an ideal containing q. In particular, $h = p \cdot f_q^{-1} g + \alpha \in A/q$, $\alpha \in Q$. Having two variables, N is chosen shorter, $n = N^2$ produce the same codeword length (and the same arithmetic cost).

GB-NTRU

These are the main differences in key creation:

- NTRU uses $A = \mathbf{Z}[x]/(x^n - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_q^{-1} g \in A/q$; q, p are public.
- GB-NTRU uses $A = \mathbf{Z}[X]/(X^N - 1)$, $q, p, f, g \in A$, and the public key is $h = p \cdot f_Q^{-1} g \in A/q$; q, p are public.

  $q \in Q \subseteq A$

  X is a pair of variables (x, y), Q is an ideal containing q. In particular, $h = p \cdot f_q^{-1} g + \alpha \in A/q$, $\alpha \in Q$. Having two variables, N is chosen shorter, $n = N^2$ produce the same codeword length (and the same arithmetic cost).
- In both, to encypher, given a message m, choose a random r and compute $c = hr + m \in A/q$

The private ideal Q and its use

Q **is part of the private key!** We have $h = p \cdot g/f \in A/Q$ (private), but the public only has $h = p \cdot g/f + \alpha \in A/q$. Hence an eavesdropper has no way to recover f, g without guessing Q. It would be like a GB-RSA for which pq is private, the public key is $pq + c$, we need to retrieve p and q, but even if we know how to factor we don't know **what** to factor.

The private Q makes the attack of Coppersmith-Shamir to the NTRU key impossible. This allows to choose smaller f, g, and this in turn allows to choose larger m, f, increasing the security of the message.

The presence of Q has of course consequences in the decyphering:

Decyphering

- In NTRU, to decypher compute $fc \in A/q$, $fc = p \cdot gh + mf \in A/q$.

Decyphering

- In NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/q$.
- In GB-NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/Q$, $fc = p \cdot gh + mf + \beta \in A/q$, $\beta \in Q$.
  To be able to continue, one has to find $p \cdot gh + mf \in A/q$

Decyphering

- In NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/q$.
- In GB-NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/Q$, $fc = p \cdot gh + mf + \beta \in A/q$, $\beta \in Q$.
  To be able to continue, one has to find $p \cdot gh + mf \in A/q$

  $Q \subseteq A$ and A as group is $\mathbf{Z}^{N^2}$: Q is a lattice, under suitable conditions β is the closest vector to fc.

Decyphering

- In NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/q$.
- In GB-NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/Q$, $fc = p \cdot gh + mf + \beta \in A/q$, $\beta \in Q$.
  To be able to continue, one has to find $p \cdot gh + mf \in A/q$

  $Q \subseteq A$ and A as group is $\mathbf{Z}^{N^2}$: Q is a lattice, under suitable conditions β is the closest vector to fc.

  If β is correctly identified, then decyphering continues.

Decyphering

- In NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/q$.
- In GB-NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/Q$, $fc = p \cdot gh + mf + \beta \in A/q$, $\beta \in Q$.
  To be able to continue, one has to find $p \cdot gh + mf \in A/q$

  $Q \subseteq A$ and A as group is $\mathbf{Z}^{N^2}$: Q is a lattice, under suitable conditions β is the closest vector to fc.

  If β is correctly identified, then decyphering continues.
- In both, under suitable conditions, lifting fc to A and reducing mod p, one recovers $fm \in A/p$, hence m.

Decyphering

- In NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/q$.
- In GB-NTRU, to decypher compute $fc \in A/q$,
  $fc = p \cdot gh + mf \in A/Q$, $fc = p \cdot gh + mf + \beta \in A/q$, $\beta \in Q$.
  To be able to continue, one has to find $p \cdot gh + mf \in A/q$

  $Q \subseteq A$ and A as group is $\mathbf{Z}^{N^2}$: Q is a lattice, under suitable conditions β is the closest vector to fc.

  If β is correctly identified, then decyphering continues.
- In both, under suitable conditions, lifting fc to A and reducing mod p, one recovers $fm \in A/p$, hence m.
- During the decyphering, one has to find not only m, but also r, to check the conformity to the specifications; otherwise chosen cyphertext attacks might disclose the private lattice.

CVP for Q

We have to solve a CVP for the lattice Q; depending on the lattice and on the vector the problem might be easy.

In our tests with reasonable parameters, for random choices of Q the CVP for fc is always easily solved via Babai closest plane algorithm, but for at least 0.1% of random Q for 99% of the messages the (much faster) round-off algorithm is enough.

We believe that the quality of Q might be corrrelated with the geometric properties of the zero-set of Q, and this might be exploited, either to build good keys, or to attack the private lattice.

Security of NTRU vs. GB-NTRU

Avoiding the Coppersmith-Shamir key attack (and other key attacks) improves the overall security of the cryptosystem. It might allow to choose smaller f and g, (increasing the size of f and g makes the private key more secure) hence one may choose larger r and m (making the message more secure).
As a consequence, this might allow to choose shorter lengths, and reduce the computational cost of decoding, compensating the increased complexity.

Conclusions. Was Barkee wrong? (Where are Gröbner bases in GB-NTRU?)

In GB-NTRU a Gröbner basis of Q is used to invert f and to perform computations mod Q. We have to admit however that other methods can be used, so there is really no GB in GB-NTRU.

In GB-GGH, aka LPC Gröbner bases are essential. We are quite confident to eventually come with a secure and relatively practical cryptosystem, but still we don't have conclusive evidence.

So up to now we consider Barkee challenge still open.
Up to now, we just **hope**.

We have a proof-of-concept implementation, not yet ready for prime time.

The work on these cryptosystems is still in progress.

More details in
Massimo Caboara, Fabrizio Caruso, Carlo Traverso
"Gröbner Bases for Public Key Cryptography",
ISSAC'08, July 20–23, 2008, Hagenberg, Austria.

(preprints in `http://posso.dm.unipi.it/crypto`)