Factoring Unbalanced Moduli with Known Bits

Éric Brier[1] David Naccache[2]
Mehdi Tibouchi[2]

[1]Ingenico

[2]École normale supérieure

ICISC 2009

Our Results in a Nutshell

- Investigate the problem of factoring an unbalanced RSA modulus $n = pq$ $(p > q)$ given the knowledge of some bits of p.
- Find that it is easily solved when at least $2\log_2 q$ contiguous bits of p are known, regardless of their position.
- Show that this bound can be improved depending on where the known bit pattern is located, and that different (e.g. non-contiguous) patterns can be tackled as well.

Our Results in a Nutshell

- Investigate the problem of factoring an unbalanced RSA modulus $n = pq$ $(p > q)$ given the knowledge of some bits of p.
- Find that it is easily solved when at least $2 \log_2 q$ contiguous bits of p are known, regardless of their position.
- Show that this bound can be improved depending on where the known bit pattern is located, and that different (e.g. non-contiguous) patterns can be tackled as well.

Our Results in a Nutshell

- Investigate the problem of factoring an unbalanced RSA modulus $n = pq$ ($p > q$) given the knowledge of some bits of p.
- Find that it is easily solved when at least $2\log_2 q$ contiguous bits of p are known, regardless of their position.
- Show that this bound can be improved depending on where the known bit pattern is located, and that different (e.g. non-contiguous) patterns can be tackled as well.

Outline

Outline

The Factoring Problem

- Factoring large integers is believed to be computationally hard, and many cryptographic primitives are based on this hardness, or the hardness of related problems (such as RSA).
- However, practical implementations of such primitives may leak some secret information on the number to be factored, and an adversary could conceivably use this information to recover the secret factors efficiently.
- It is thus interesting to investigate how resilient the factoring problem is to this sort of leakage. Line of work initiated by Rivest and Shamir in 1986.

The Factoring Problem

- Factoring large integers is believed to be computationally hard, and many cryptographic primitives are based on this hardness, or the hardness of related problems (such as RSA).
- However, practical implementations of such primitives may leak some secret information on the number to be factored, and an adversary could conceivably use this information to recover the secret factors efficiently.
- It is thus interesting to investigate how resilient the factoring problem is to this sort of leakage. Line of work initiated by Rivest and Shamir in 1986.

The Factoring Problem

- Factoring large integers is believed to be computationally hard, and many cryptographic primitives are based on this hardness, or the hardness of related problems (such as RSA).
- However, practical implementations of such primitives may leak some secret information on the number to be factored, and an adversary could conceivably use this information to recover the secret factors efficiently.
- It is thus interesting to investigate how resilient the factoring problem is to this sort of leakage. Line of work initiated by Rivest and Shamir in 1986.

Factoring with a hint

- The original 1986 paper by Rivest and Shamir shows that one can factor a balanced RSA modulus $n = pq$ of N bits given the knowledge of the $N/3$ top (or bottom) bits of p. In 1995, Coppersmith improved this result to $N/4$.
- These results have been extended in various directions (such as numbers with more than two prime factors, or numbers of the form $p^r q$), usually using a chunk of bits as the leaked information on the factors.
- Other types of hints have also been considered, such as an oracle answering arbitrary yes/no questions, or an oracle returning another composite integer whose factorization is related to the initial one.

Factoring with a hint

- The original 1986 paper by Rivest and Shamir shows that one can factor a balanced RSA modulus $n = pq$ of N bits given the knowledge of the $N/3$ top (or bottom) bits of p. In 1995, Coppersmith improved this result to $N/4$.
- These results have been extended in various directions (such as numbers with more than two prime factors, or numbers of the form $p^r q$), usually using a chunk of bits as the leaked information on the factors.
- Other types of hints have also been considered, such as an oracle answering arbitrary yes/no questions, or an oracle returning another composite integer whose factorization is related to the initial one.

Factoring with a hint

- The original 1986 paper by Rivest and Shamir shows that one can factor a balanced RSA modulus $n = pq$ of N bits given the knowledge of the $N/3$ top (or bottom) bits of p. In 1995, Coppersmith improved this result to $N/4$.
- These results have been extended in various directions (such as numbers with more than two prime factors, or numbers of the form $p^r q$), usually using a chunk of bits as the leaked information on the factors.
- Other types of hints have also been considered, such as an oracle answering arbitrary yes/no questions, or an oracle returning another composite integer whose factorization is related to the initial one.

Coppersmith's method I

- Most recent results on factoring with hints (and ours is no exception) use Coppersmith's lattice-based techniques for finding small roots of polynomial equations.
- We give a quick run-down of Coppersmith's own 1995 method for factoring with hints.
- Let $n = pq$ the number to be factored, and say the top L bits of p are known to an attacker. He then deduces the top L bits of q by division, and obtains an equation of the form $n = (p_0 + x)(q_0 + y)$, i.e.:

$$xy + q_0x + p_0y + (n - p_0q_0) = 0$$

where p_0, q_0 are known constants and x, y are the unknown bit patterns.

Coppersmith's method I

- Most recent results on factoring with hints (and ours is no exception) use Coppersmith's lattice-based techniques for finding small roots of polynomial equations.
- We give a quick run-down of Coppersmith's own 1995 method for factoring with hints.
- Let $n = pq$ the number to be factored, and say the top L bits of p are known to an attacker. He then deduces the top L bits of q by division, and obtains an equation of the form $n = (p_0 + x)(q_0 + y)$, i.e.:

$$xy + q_0x + p_0y + (n - p_0q_0) = 0$$

where p_0, q_0 are known constants and x, y are the unknown bit patterns.

Coppersmith's method I

- Most recent results on factoring with hints (and ours is no exception) use Coppersmith's lattice-based techniques for finding small roots of polynomial equations.
- We give a quick run-down of Coppersmith's own 1995 method for factoring with hints.
- Let $n = pq$ the number to be factored, and say the top L bits of p are known to an attacker. He then deduces the top L bits of q by division, and obtains an equation of the form $n = (p_0 + x)(q_0 + y)$, i.e.:

$$xy + q_0 x + p_0 y + (n - p_0 q_0) = 0$$

where p_0, q_0 are known constants and x, y are the unknown bit patterns.

Coppersmith's method I

- Most recent results on factoring with hints (and ours is no exception) use Coppersmith's lattice-based techniques for finding small roots of polynomial equations.
- We give a quick run-down of Coppersmith's own 1995 method for factoring with hints.
- Let $n = pq$ the number to be factored, and say the top L bits of p are known to an attacker. He then deduces the top L bits of q by division, and obtains an equation of the form $n = (p_0 + x)(q_0 + y)$, i.e.:

$$xy + q_0 x + p_0 y + (n - p_0 q_0) = 0$$

where p_0, q_0 are known constants and x, y are the unknown bit patterns.

Coppersmith's method I

- Most recent results on factoring with hints (and ours is no exception) use Coppersmith's lattice-based techniques for finding small roots of polynomial equations.
- We give a quick run-down of Coppersmith's own 1995 method for factoring with hints.
- Let $n = pq$ the number to be factored, and say the top L bits of p are known to an attacker. He then deduces the top L bits of q by division, and obtains an equation of the form $n = (p_0 + x)(q_0 + y)$, i.e.:

$$xy + q_0 x + p_0 y + (n - p_0 q_0) = 0$$

  where p_0, q_0 are known constants and x, y are the unknown bit patterns.

Coppersmith's method II

- The previous equation:

$$p(x, y) = xy + q_0 x + p_0 y + (n - p_0 q_0) = 0$$

is a bivariate equation of degree 2 over the integers, with a small root (x, y).

- Coppersmith shows that it can be solved provided that the bitsizes of X and Y satisfy $X + Y < 2D/3$, where D is the maximum bitsize of $p(x, y)$ in the required range, in this case $N/2 + X$ for a balanced modulus with $X = Y$.
- Hence the size of the unknown chunk must satisfy:

$$6X < N + 2X \quad \text{i.e.} \quad X < N/4$$

as required.

Coppersmith's method II

- The previous equation:

$$p(x, y) = xy + q_0 x + p_0 y + (n - p_0 q_0) = 0$$

  is a bivariate equation of degree 2 over the integers, with a small root (x, y).

- Coppersmith shows that it can be solved provided that the bitsizes of X and Y satisfy $X + Y < 2D/3$, where D is the maximum bitsize of $p(x, y)$ in the required range, in this case $N/2 + X$ for a balanced modulus with $X = Y$.

- Hence the size of the unknown chunk must satisfy:

$$6X < N + 2X \quad \text{i.e.} \quad X < N/4$$

  as required.

Coppersmith's method II

- The previous equation:

$$p(x, y) = xy + q_0 x + p_0 y + (n - p_0 q_0) = 0$$

  is a bivariate equation of degree 2 over the integers, with a small root (x, y).
- Coppersmith shows that it can be solved provided that the bitsizes of X and Y satisfy $X + Y < 2D/3$, where D is the maximum bitsize of $p(x, y)$ in the required range, in this case $N/2 + X$ for a balanced modulus with $X = Y$.
- Hence the size of the unknown chunk must satisfy:

$$6X < N + 2X \quad \text{i.e.} \quad X < N/4$$

  as required.

Outline

Unbalanced moduli

- Unbalanced moduli $n = pq$, with p much larger than q, aren't as commonly used in cryptography as balanced ones.
- However, several proposed schemes use such moduli. For instance, in his 1990 "RSA for paranoids" paper, Shamir showed how RSA security could be improved at little computational cost by choosing q of regular RSA size but p much larger.
- This doesn't improve security as much as a larger balanced n would (because of factoring algorithms such as the ECM), but it performs a lot faster.
- However, a larger n means a larger key size, and a larger amount of secret data to protect from leakage.

Unbalanced moduli

- Unbalanced moduli $n = pq$, with p much larger than q, aren't as commonly used in cryptography as balanced ones.
- However, several proposed schemes use such moduli. For instance, in his 1990 "RSA for paranoids" paper, Shamir showed how RSA security could be improved at little computational cost by choosing q of regular RSA size but p much larger.
- This doesn't improve security as much as a larger balanced n would (because of factoring algorithms such as the ECM), but it performs a lot faster.
- However, a larger n means a larger key size, and a larger amount of secret data to protect from leakage.

Unbalanced moduli

- Unbalanced moduli $n = pq$, with p much larger than q, aren't as commonly used in cryptography as balanced ones.
- However, several proposed schemes use such moduli. For instance, in his 1990 "RSA for paranoids" paper, Shamir showed how RSA security could be improved at little computational cost by choosing q of regular RSA size but p much larger.
- This doesn't improve security as much as a larger balanced n would (because of factoring algorithms such as the ECM), but it performs a lot faster.
- However, a larger n means a larger key size, and a larger amount of secret data to protect from leakage.

Unbalanced moduli

- Unbalanced moduli $n = pq$, with p much larger than q, aren't as commonly used in cryptography as balanced ones.
- However, several proposed schemes use such moduli. For instance, in his 1990 "RSA for paranoids" paper, Shamir showed how RSA security could be improved at little computational cost by choosing q of regular RSA size but p much larger.
- This doesn't improve security as much as a larger balanced n would (because of factoring algorithms such as the ECM), but it performs a lot faster.
- However, a larger n means a larger key size, and a larger amount of secret data to protect from leakage.

Outline

Some trivial cases

- General situation: we want to factor an unbalanced modulus $n = pq$ where q is of bit-length Q, under the assumption that some L-bit chunk of p is known.
- A trivial case is when the Q most significant bits of p are known. Indeed, a simple division recovers all Q bits of q.
- Similarly, if the Q least significant bits of p are known, say $p' = p \bmod 2^Q$, we can compute:

$$\frac{n}{p'} \bmod 2^Q = q \bmod 2^Q = q$$

Some trivial cases

- General situation: we want to factor an unbalanced modulus $n = pq$ where q is of bit-length Q, under the assumption that some L-bit chunk of p is known.
- A trivial case is when the Q most significant bits of p are known. Indeed, a simple division recovers all Q bits of q.
- Similarly, if the Q least significant bits of p are known, say $p' = p \bmod 2^Q$, we can compute:

$$\frac{n}{p'} \bmod 2^Q = q \bmod 2^Q = q$$

Some trivial cases

- General situation: we want to factor an unbalanced modulus $n = pq$ where q is of bit-length Q, under the assumption that some L-bit chunk of p is known.
- A trivial case is when the Q most significant bits of p are known. Indeed, a simple division recovers all Q bits of q.
- Similarly, if the Q least significant bits of p are known, say $p' = p \bmod 2^Q$, we can compute:

$$\frac{n}{p'} \bmod 2^Q = q \bmod 2^Q = q$$

Some trivial cases

- General situation: we want to factor an unbalanced modulus $n = pq$ where q is of bit-length Q, under the assumption that some L-bit chunk of p is known.
- A trivial case is when the Q most significant bits of p are known. Indeed, a simple division recovers all Q bits of q.
- Similarly, if the Q least significant bits of p are known, say $p' = p \bmod 2^Q$, we can compute:

$$\frac{n}{p'} \bmod 2^Q = q \bmod 2^Q = q$$

A simple pattern elsewhere

- While a bit pattern at either end of p readily provides information on q, it is not as easy to take advantage of a bit pattern elsewhere.
- One simple example is the case when p has a Q-bit chunk of zero bits starting from position Q.
- Indeed, this gives $p = 2^{2Q} + y$ where y is of bitsize Q. We can thus write:

$$\gcd(n, n \bmod 2^{2Q}) = \gcd(pq, yq \bmod 2^{2Q}) = \gcd(pq, yq) = q$$

A simple pattern elsewhere

- While a bit pattern at either end of p readily provides information on q, it is not as easy to take advantage of a bit pattern elsewhere.
- One simple example is the case when p has a Q-bit chunk of zero bits starting from position Q.

- Indeed, this gives $p = 2^{2Q} + y$ where y is of bitsize Q. We can thus write:

$$\gcd(n, n \bmod 2^{2Q}) = \gcd(pq, yq \bmod 2^{2Q}) = \gcd(pq, yq) = q$$

A simple pattern elsewhere

- While a bit pattern at either end of p readily provides information on q, it is not as easy to take advantage of a bit pattern elsewhere.
- One simple example is the case when p has a Q-bit chunk of zero bits starting from position Q.

- Indeed, this gives $p = 2^{2Q} + y$ where y is of bitsize Q. We can thus write:

$$\gcd(n, n \bmod 2^{2Q}) = \gcd(pq, yq \bmod 2^{2Q}) = \gcd(pq, yq) = q$$

Tackling more general patterns

- The previous case $p = 2^{2Q} + y$ generalizes to:

$$p = u \cdot 2^{W+L} + v \cdot 2^{W} + y$$

with u, y unknown, and v a known L-bit pattern starting from position W.

- To take advantage of our knowledge of v, we reduce the equation $n = pq \bmod 2^{W+L}$:

$$(n \bmod \mathrm{mod} 2^{W+L}) = (v \cdot 2^{W} + y) \cdot q \pmod 2^{W+L}$$

- This has the form:

$$b = x(a + y) \pmod 2^{W+L}$$

where x and y are unknowns of bitsizes Q and W respectively, whereas a and b are known constants.

- This equation is then well-suited for applying techniques based on lattice reduction.

Tackling more general patterns

- The previous case $p = 2^{2Q} + y$ generalizes to:

$$p = u \cdot 2^{W+L} + v \cdot 2^{W} + y$$

  with u, y unknown, and v a known L-bit pattern starting from position W.

- To take advantage of our knowledge of v, we reduce the equation $n = pq \bmod 2^{W+L}$:

$$(n \bmod \bmod 2^{W+L}) = (v \cdot 2^{W} + y) \cdot q \pmod 2^{W+L}$$

- This has the form:

$$b = x(a + y) \pmod 2^{W+L}$$

  where x and y are unknowns of bitsizes Q and W respectively, whereas a and b are known constants.

- This equation is then well-suited for applying techniques based on lattice reduction.

Tackling more general patterns

- The previous case $p = 2^{2Q} + y$ generalizes to:

$$p = u \cdot 2^{W+L} + v \cdot 2^{W} + y$$

with u, y unknown, and v a known L-bit pattern starting from position W.

- To take advantage of our knowledge of v, we reduce the equation $n = pq \bmod 2^{W+L}$:

$$(n \bmod \bmod 2^{W+L}) = (v \cdot 2^{W} + y) \cdot q \pmod 2)^{W+L}$$

- This has the form:

$$b = x(a + y) \pmod 2)^{W+L}$$

where x and y are unknowns of bitsizes Q and W respectively, whereas a and b are known constants.

- This equation is then well-suited for applying techniques based on lattice reduction.

Tackling more general patterns

- The previous case $p = 2^{2Q} + y$ generalizes to:

$$p = u \cdot 2^{W+L} + v \cdot 2^{W} + y$$

with u, y unknown, and v a known L-bit pattern starting from position W.

- To take advantage of our knowledge of v, we reduce the equation $n = pq \bmod 2^{W+L}$:

$$(n \bmod \bmod 2^{W+L}) = (v \cdot 2^{W} + y) \cdot q \pmod 2^{W+L}$$

- This has the form:

$$b = x(a + y) \pmod 2^{W+L}$$

where x and y are unknowns of bitsizes Q and W respectively, whereas a and b are known constants.

- This equation is then well-suited for applying techniques based on lattice reduction.

Outline

Linearization

- To recover x and y in the previous equation:

$$b = x(a + y) \pmod{2}^{W+L}$$

a first, simple method is linearization: write $z = xy$. The equation simplifies to a modular linear equation in two variables:

$$b = ax + z \pmod{2}^{W+L}$$

- The solutions (x, z) form a lattice in $\mathbb{Z}^2$. Lattice reduction algorithms like LLL will thus recover the solution we are after if it is small enough.
- In our setting, x is of size Q and z of size $Q + W$. Therefore, LLL will work provided that $Q + (Q + W) < W + L$, i.e. $L > 2Q$.

Linearization

- To recover x and y in the previous equation:

$$b = x(a + y) \pmod{2}^{W+L}$$

  a first, simple method is linearization: write $z = xy$. The equation simplifies to a modular linear equation in two variables:

$$b = ax + z \pmod{2}^{W+L}$$

- The solutions (x, z) form a lattice in $\mathbb{Z}^2$. Lattice reduction algorithms like LLL will thus recover the solution we are after if it is small enough.
- In our setting, x is of size Q and z of size $Q + W$. Therefore, LLL will work provided that $Q + (Q + W) < W + L$, i.e. $L > 2Q$.

Linearization

- To recover x and y in the previous equation:

$$b = x(a + y) \pmod{2}^{W+L}$$

a first, simple method is linearization: write $z = xy$. The equation simplifies to a modular linear equation in two variables:

$$b = ax + z \pmod{2}^{W+L}$$

- The solutions (x, z) form a lattice in $\mathbb{Z}^2$. Lattice reduction algorithms like LLL will thus recover the solution we are after if it is small enough.
- In our setting, x is of size Q and z of size $Q + W$. Therefore, LLL will work provided that $Q + (Q + W) < W + L$, i.e. $L > 2Q$.

Linearization

- To recover x and y in the previous equation:

$$b = x(a + y) \pmod{2}^{W+L}$$

a first, simple method is linearization: write $z = xy$. The equation simplifies to a modular linear equation in two variables:

$$b = ax + z \pmod{2}^{W+L}$$

- The solutions (x, z) form a lattice in $\mathbb{Z}^2$. Lattice reduction algorithms like LLL will thus recover the solution we are after if it is small enough.
- In our setting, x is of size Q and z of size $Q + W$. Therefore, LLL will work provided that $Q + (Q + W) < W + L$, i.e. $L > 2Q$.

Linearization

In other words, a $2Q$-bit pattern anywhere in p is enough to factor n.

Better bound with Coppersmith

- Instead of linearizing the equation

$$b = x(a + y) \pmod{2}^{W+L}$$

  we can try to solve it directly.
- As a bivariate modular quadratic, this equation can be tackled with extensions of Coppersmith's method for recovering small roots of polynomial equations.
- While the original Coppersmith theorems apply to either univariate modular polynomials or bivariate polynomials over $\mathbb{Z}$, they generalize heuristically to the multivariate modular case, subject to appropriate bounds given by Howgrave-Graham.
- Moreover, this particular quadratic equation is well understood: it is a simple variant of the Boneh-Durfee equation $x(a + y) = 1 \bmod e$, whose root can be recovered if x and y satisfy bounds that are easy to express.

Better bound with Coppersmith

- Instead of linearizing the equation

$$b = x(a+y) \pmod{2}^{W+L}$$

we can try to solve it directly.
- As a bivariate modular quadratic, this equation can be tackled with extensions of Coppersmith's method for recovering small roots of polynomial equations.
- While the original Coppersmith theorems apply to either univariate modular polynomials or bivariate polynomials over $\mathbb{Z}$, they generalize heuristically to the multivariate modular case, subject to appropriate bounds given by Howgrave-Graham.
- Moreover, this particular quadratic equation is well understood: it is a simple variant of the Boneh-Durfee equation $x(a+y) = 1 \bmod e$, whose root can be recovered if x and y satisfy bounds that are easy to express.

Better bound with Coppersmith

- Instead of linearizing the equation

$$b = x(a+y) \pmod{2}^{W+L}$$

we can try to solve it directly.

- As a bivariate modular quadratic, this equation can be tackled with extensions of Coppersmith's method for recovering small roots of polynomial equations.
- While the original Coppersmith theorems apply to either univariate modular polynomials or bivariate polynomials over $\mathbb{Z}$, they generalize heuristically to the multivariate modular case, subject to appropriate bounds given by Howgrave-Graham.
- Moreover, this particular quadratic equation is well understood: it is a simple variant of the Boneh-Durfee equation $x(a+y) = 1 \bmod e$, whose root can be recovered if x and y satisfy bounds that are easy to express.

Better bound with Coppersmith

- Instead of linearizing the equation

$$b = x(a + y) \pmod{2}^{W+L}$$

  we can try to solve it directly.
- As a bivariate modular quadratic, this equation can be tackled with extensions of Coppersmith's method for recovering small roots of polynomial equations.
- While the original Coppersmith theorems apply to either univariate modular polynomials or bivariate polynomials over $\mathbb{Z}$, they generalize heuristically to the multivariate modular case, subject to appropriate bounds given by Howgrave-Graham.
- Moreover, this particular quadratic equation is well understood: it is a simple variant of the Boneh-Durfee equation $x(a + y) = 1 \bmod e$, whose root can be recovered if x and y satisfy bounds that are easy to express.

Better bound with Coppersmith

- The lattice involved in solving our equation has the same form and the same determinant as Boneh and Durfee's, so the bound on x and y is computed identically.
- However, contrary to the Boneh-Durfee setting, x and y need not have the same size in our case. Adapting the computation, we find that (x, y) can be recovered iff:

$$L > Q + \frac{2}{3}(\sqrt{W^2 + 3QW} - W)$$

- The number of known bits L required to factor n using this method is thus close to Q for small W, and increases asymptotically to $2Q$ for $W \to \infty$.

Better bound with Coppersmith

- The lattice involved in solving our equation has the same form and the same determinant as Boneh and Durfee's, so the bound on x and y is computed identically.
- However, contrary to the Boneh-Durfee setting, x and y need not have the same size in our case. Adapting the computation, we find that (x, y) can be recovered iff:

$$L > Q + \frac{2}{3}(\sqrt{W^2 + 3QW} - W)$$

- The number of known bits L required to factor n using this method is thus close to Q for small W, and increases asymptotically to $2Q$ for $W \to \infty$.

Better bound with Coppersmith

- The lattice involved in solving our equation has the same form and the same determinant as Boneh and Durfee's, so the bound on x and y is computed identically.
- However, contrary to the Boneh-Durfee setting, x and y need not have the same size in our case. Adapting the computation, we find that (x, y) can be recovered iff:

$$L > Q + \frac{2}{3}(\sqrt{W^2 + 3QW} - W)$$

- The number of known bits L required to factor n using this method is thus close to Q for small W, and increases asymptotically to $2Q$ for $W \to \infty$.

Better bound with Coppersmith

Required number of known bits growing from Q to $2Q$ as the chunk slides from the least significant bits to the most significant bits. The method is always better than linearization.

Outline

Other patterns

- It is also possible to use hints consisting of multiple non-contiguous bit blocks of p to factor n. In the paper, we consider the case when the the least significant $Q/2$ bits of p are known, as well as a Q-bit pattern starting from position Q. This data is sufficient to factor.

- Furthermore, particular forms of n itself can further improve the number of bits needed to factor. We find that a short, suitably placed string of zeroes in n can improve the bounds of our Coppersmith-type technique by about 5%.

Other patterns

- It is also possible to use hints consisting of multiple non-contiguous bit blocks of p to factor n. In the paper, we consider the case when the the least significant $Q/2$ bits of p are known, as well as a Q-bit pattern starting from position Q. This data is sufficient to factor.

- Furthermore, particular forms of n itself can further improve the number of bits needed to factor. We find that a short, suitably placed string of zeroes in n can improve the bounds of our Coppersmith-type technique by about 5%.

Conclusion

- At most $2Q$ contiguous bits of p, appearing anywhere, are needed to factor some unbalanced modulus pq.
- In many cases, even fewer bits are needed.
- Additional patterns suggest that further improvements and generalizations are possible.

Conclusion

- At most $2Q$ contiguous bits of p, appearing anywhere, are needed to factor some unbalanced modulus pq.
- In many cases, even fewer bits are needed.
- Additional patterns suggest that further improvements and generalizations are possible.

Conclusion

- At most $2Q$ contiguous bits of p, appearing anywhere, are needed to factor some unbalanced modulus pq.
- In many cases, even fewer bits are needed.
- Additional patterns suggest that further improvements and generalizations are possible.

Thank you!