# A New Related Message Attack on RSA

Oded Yacobi[1] and Yacov Yacobi[2]

[1] Department of Mathematics, University of California San Diego,
9500 Gilman Drive, La Jolla, CA 92093, USA
oyacobi@math.ucsd.edu,

[2] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
yacov@microsoft.com

**Abstract.** Coppersmith, Franklin, Patarin, and Reiter show that given two RSA cryptograms $x^e \bmod N$ and $(ax+b)^e \bmod N$ for known constants $a, b \in \mathbb{Z}_N$, one can compute $x$ in $O(e \log^2 e)$ $\mathbb{Z}_N$-operations with some positive error probability. We show that given $e$ cryptograms $c_i \equiv (a_i x + b_i)^e \bmod N$, $i = 0, 1, ...e-1$, for any known constants $a_i, b_i \in \mathbb{Z}_N$, one can deterministically compute $x$ in $O(e)$ $\mathbb{Z}_N$-operations that depend on the cryptograms, after a pre-processing that depends only on the constants. The complexity of the pre-processing is $O(e \log^2 e)$ $\mathbb{Z}_N$-operations, and can be amortized over many instances. We also consider a special case where the overall cost of the attack is $O(e)$ $\mathbb{Z}_N$-operations. Our tools are borrowed from numerical-analysis and adapted to handle formal polynomials over finite-rings. To the best of our knowledge their use in cryptanalysis is novel.

## 1 Introduction

Messages with known relations may occur for example if an attacker pretends to be the recipient in a protocol that doesn't authenticate the recipient, and in addition the message is composed of the content concatenated with a serial number. In that case the attacker can claim that she didn't receive the transmission properly and ask that it be sent again. The next transmission will have the same content as the original but an incremented serial number. If the increment is known we have a known relation. Other examples appear in [4].

Related message attacks can be avoided all together if before RSA-encryption the message $M$ is transformed using e.g. the OAEP function ([3]; There are other methods and some issues are not settled yet, see [5]). This transformation destroys the relations between messages and increases the message length.

Nevertheless it is useful to know the ramifications in case for some reason one chooses not to use OAEP or similar methods (even though it is highly recommended). For example RFID tags may pose tough engineering challenges of creating very compact cryptosystems, and the trade-off must be known precisely.

In [4] it was shown that given two RSA cryptograms $x^e \bmod N$, and $(ax+b)^e \bmod N$ for any known constants $a, b \in \mathbb{Z}_N$ one can compute $x$ in $O(e \log^2 e)$ $\mathbb{Z}_N$-operations with some small error probability.

We show that given $e$ cryptograms $c_i \equiv (a_i x + b_i)^e \bmod N$, $i = 0, 1, ...e-1$, for any known constants $a_i, b_i \in \mathbb{Z}_N$, one can deterministically compute $x$ in $O(e)$ $\mathbb{Z}_N$-operations, after doing $O(e \log^2 e)$ pre-computations that depend only on the known constants. The descriptions of the protocol and the attack determine the values of these constants. For example the attack described at the beginning of this section has for all $i$ $a_i = b_i = 1$. The cost of the pre-computations can be amortized over many instances of the problem.

Our problem could be solved by using the Newton expansion of $c_i \equiv (a_i x + b_i)^e \bmod N$, renaming $z_j = x^j$ and using linear algebra to find $z_1$. However, our method is more efficient.

We also show that in the special case where $c_i \equiv (ax + b \cdot i)^e \bmod N$, $i = 0, 1, ...e-1$, for any known constants $a, b \in \mathbb{Z}_N$, where $\gcd(a, N) = \gcd(b, N) = \gcd(e!, N) = 1$, one can deterministically compute $x$ in overall $O(e)$ $\mathbb{Z}_N$-operations using

$$x \equiv a^{-1} b[(b^e e!)^{-1} \sum_{i=0}^{e-1} \binom{e-1}{i} \cdot c_i \cdot (-1)^{e-1+i} - \frac{e-1}{2}] \bmod N$$

If any of the above gcd conditions do not hold then the system is already broken.

It remains an open problem whether the new approach can improve the general case of implicit linear dependence, i.e., suppose for known constants $a_i$, $i = 0, 1, 2, ...k$, there is a known relation $\sum_{i=1}^{k} a_i x_i = a_0$ among messages $x_1, x_2, ...x_k$. The current complexity of attacking this problem is $O(e^{k/2} k^2)$ [4].

Our major attack-tools are divided-differences and finite-differences. These tools are borrowed from numerical-analysis, and adapted to handle formal polynomials over finite-rings. To the best of our knowledge their use in cryptanalysis is novel.

For a survey of the work on breaking RSA see [2].

## 2 Main Result

### 2.1 Divided Differences

We borrow the concept of *divided-differences* from numerical analysis and adapt it to handle formal polynomials over finite rings. This will allow us to extract the message from a string of $e$ cryptograms whose underlying messages are linearly related. We specialize our definitions to the ring of integers modulo $N$, a product of two primes (the "RSA ring"). All the congruences in this paper are taken mudulo $N$.

**Definition 1.** *Let $h$ be a polynomial defined over the ring of integers modulo $N$, and let $x_0, x_1, ...x_n$ be distinct elements of the ring such that $(x_0 - x_i)^{-1} \bmod N$ exist for $i = 0, 1, ...n$. The $n^{th}$ divided-difference of $h$ relative to these elements is defined as follows:*

$[x_i] \equiv h(x_i)$,
$[x_0, x_1] \equiv \frac{[x_0]-[x_1]}{x_0-x_1}$,
$[x_0, x_1, ....x_n] \equiv \frac{[x_0,x_1,...x_{n-1}]-[x_1,x_2,...x_n]}{x_0-x_n}$.

Let $x$ be an indeterminate variable, and for $i = 0, 1, ...n$, let $x_i \equiv x + b_i$ for some known constants $b_i$ (these are the general explicit linear relations that we assume later). We can now view the above divided differences as univariate polynomials in $x$ defined over $\mathbb{Z}_N$.

The following lemma is true for the divided difference of any polynomial $\bmod N$, but for our purposes it is enough to prove it for the RSA polynomial $x^e \bmod N$. Related results are stated in [8]. Before beginning the proof we introduce some notation borrowed from [7]. Let $\pi_k(y) \equiv \prod_{i=0}^{k}(y - x_i)$. Then taking the derivative of $\pi_k$ with respect to $y$ we have for $i \le k$

$$\pi'_k(x_i) \equiv \prod_{\substack{0 \le j \le k \\ j \ne i}} (x_i - x_j)$$

By induction on $k$ the following equality easily follows

$$[x_0, ..., x_k] \equiv \sum_{i=0}^{k} \frac{h(x_i)}{\pi'_k(x_i)} \tag{1}$$

Let $C_t(p)$ denote the $t_{th}$ coefficient of the polynomial $p$, starting from the leading coefficients (the coefficients of the highest powers). We use $C_t[x_0, ..x_k]$ as a shorthand for $C_t([x_0, ..x_k])$.

**Lemma 1.** *Let $[x_0, ..., x_n]$ be the $n^{th}$ divided difference relative to the RSA polynomial $h(x) \equiv x^e \bmod N$, and let $x_0, x_1, ...x_n$ be distinct elements of the ring such that $(x_0 - x_i)^{-1} \bmod N$ exist for $i = 0, 1, ...n$. Then (i) for $0 \le n \le e$, if $\binom{e}{e-n} \ne 0 \bmod N$ then $\deg[x_0, ..., x_n] = e - n$. (ii) $C_{e-n}[x_0, x_1, .., x_n] \equiv \binom{e}{e-n}$ (an important special case is $C_1[x_0, x_1, .., x_{e-1}] \equiv e \bmod N$).*

Comment: In practice the condition in claim (i) always holds, since $e << N$.

*Proof.* The claim is trivial for $n = 0$. For $n \ge 1$ we prove the equivalent proposition that $C_t\,[x_0, ..., x_n] = 0$ for $t = e, e-1, ..., e-n+1$ and $C_{e-n}[x_0, ..., x_n]$ is independent of the $b_i$ and is not congruent to 0. We use the notations $1/b$ and $b^{-1}$ interchangeably. We induct on $n$. When $n = 1$

$$[x_0, x_1] \equiv \frac{(x + b_0)^e - (x + b_1)^e}{b_0 - b_1} \equiv \frac{\sum_{i=0}^{e} \binom{e}{i} x^i [b_0^{e-i} - b_1^{e-i}]}{b_0 - b_1}$$

Note that by our assumption $(b_0 - b_1)^{-1} \bmod N$ exist. So $C_e[x_0, x_1] \equiv 0$ and $C_{e-1}[x_0, x_1] \equiv e$ and indeed our claim is true for $n = 1$. For the inductive hypothesis let $n = k-1$ and assume that $C_t\ [x_0,...,x_{k-1}] \equiv 0$ for $t = e, e-1, ..., e-(k-1)+1$ and $C_{e-(k-1)}[x_0,...,x_{k-1}]$ is independent of the $b_i$ and is not congruent to 0. We want to show that when $n = k$, $C_t[x_0,...,x_k] \equiv 0$ for $t = e, e-1, ..., e-k+1$ and $C_{e-k}[x_0,...,x_k]$ is independent of the $b_i$ and is not congruent to 0.

The fact that $C_t\ [x_0,...,x_k] \equiv 0$ for $t = e, e-1, ..., e-k+1$ follows immediately from the inductive hypothesis and Definition 1. It takes a little more work to show that $C_{e-k}[x_0,...,x_k]$ is independent of the $b_i$.

Using (1):

$$[x_0, x_1, ..., x_k] \equiv \sum_{i=0}^{k} \frac{(x+b_i)^e}{\pi'_k(x_i)} \equiv \sum_{j=0}^{e} \binom{e}{j} x^j [\frac{b_0^{e-j}}{\pi'_k(x_0)} + \frac{b_1^{e-j}}{\pi'_k(x_1)} + ... + \frac{b_k^{e-j}}{\pi'_k(x_k)}]$$

We want to show that $C_{e-k}[x_0, x_1, ..., x_k]$ is independent of the $b_i$.

$$C_{e-k}[x_0, x_1, .., x_k] \equiv \binom{e}{e-k} [\frac{b_0^k}{\pi'_k(x_0)} + \frac{b_1^k}{\pi'_k(x_1)} + ... + \frac{b_k^k}{\pi'_k(x_k)}] \tag{2}$$

So now it is sufficient to show that

$$(-1)^0 \frac{b_0^k}{(b_0-b_1)\cdots(b_0-b_k)} + ... + (-1)^k \frac{b_k^k}{(b_0-b_k)\cdots(b_{k-1}-b_k)} \tag{3}$$

is independent of the $b_i$.

We first multiply (3) by the necessary terms to get a common denominator. We introduce some compact notation that will simplify the process. For a given set of constants $b_0, b_1, ...b_k$ define

$$\begin{aligned} \delta(h,i) &\equiv (b_h - b_i) \\ \delta(h,i,j) &\equiv (b_h - b_i)(b_h - b_j)\delta(i,j) \\ &\vdots \\ \delta(i_0,...,i_k) &\equiv (b_{i_0} - b_{i_1})(b_{i_0} - b_{i_2}) \cdots (b_{i_0} - b_{i_k})\delta(i_1,...,i_k) \end{aligned}$$

Similarly we can also define $\delta_j \equiv \delta(0, 1, ..., \overline{j}, ..., k)$ where the bar denotes that the index is missing (so if $k = 4$ then $\delta_3 = \delta(0, 1, 2, 4, ))$. Then (3) becomes:

$$\frac{b_0^k\delta_0 - b_1^k\delta_1 + \cdots + (-1)^k b_k^k \delta_k}{\delta(0, 1, ..., k)} \tag{4}$$

We want to show that (4) is independent of the $b_i$. In fact it equals 1. To see this consider the Vandermonde matrix:

$$V \equiv \begin{bmatrix} 1 & b_0 & b_0^2 & \cdots & b_0^k \\ 1 & b_1 & b_1^2 & \cdots & b_1^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & b_k & b_k^2 & \cdots & b_k^k \end{bmatrix}$$

We conclude from (2) that $C_{e-k}[x_0, x_1, .., x_k] \equiv \binom{e}{e-k}$, which is certainly independent of the $b_i$. This also implies that $C_{e-k}[x_0, x_1, .., x_k]$ is not congruent to 0 when $k \leq e$. By induction we are done.

### 2.2 Related-Messages Attack

Here we consider the general case where for $i = 0, 1, ...e-1$, $x_i \equiv a_i x + b_i \bmod N$. $N = pq$ is an RSA composite ($p$ and $q$ are large primes, with some additional restrictions which are irrelevant in the current discussion), and the constants $a_i, b_i$ are known. Of course it is sufficient to consider just the case where $x_i \equiv x + b_i$. We now show how to deterministically compute $x$ in $O(e)$ $\mathbb{Z}_N$-operations after some pre-computation that depends only on the known constants. If the constants $b_i$ hold for many unknown values of cryptograms $x^e$ then the cost of pre-computations can be amortized and discarded. We show that the cost of the additional computations that depend on the value of $x$ is $O(e)$.

Specifically, $\pi'_n(x_k)$ is independent of $y$ and of $x$, hence for all $k$ these coefficients can be computed in advance. In that case the cost of computing $[x_0, x_1, ...x_{e-1}] \equiv ux + v \equiv w(x)$ is $O(e)$.

For each particular value $x$ we know how to compute the value $w(x)$ without knowing $x$ using Lemma 1 and Formula (1). More explicitly, Let $c_i \equiv (x + b_i)^e \bmod N$, $i = 0, 1, 2, ...e-1$, be the given cryptograms, whose underlying messages are linearly related, and let $\pi'_{e-1}(x_k) \equiv \prod_{\substack{i=0 \\ i \neq k}}^{e-1} (b_k - b_i)$. We use $p_k$ as a shorthand for $\pi'_{e-1}(x_k)$. Then

$$w(x) \equiv \sum_{k=0}^{e-1} \frac{[x_k]}{\pi'_{e-1}(x_k)} \equiv \sum_{k=0}^{e-1} \frac{c_k}{p_k}.$$

Here we assume that the inverses $(b_k - b_i)^{-1} \bmod N$ exist. Note that if for some $k, i$ this isn't true then we can factor the RSA-modulus $N$, by computing $\gcd(N, (b_k - b_i))$.

From Lemma 1 (ii) we know that $u = e$. Note also that $w(0) \equiv v \equiv \sum_{k=0}^{e-1} b_k^e \cdot p_k^{-1} \bmod N$, and we can compute it in the pre-computation phase (before intercepting the cryptograms). So we can find $x \equiv (w(x) - v)e^{-1} \bmod N$.

The following algorithm summarizes the above discussion:

Algorithm 1:

Given cryptograms $c_i \equiv (x + b_i)^e \bmod N,\ i = 0, 1, 2, ...e - 1,$ with known constants $b_i$, find $x$.

Method:

1. Pre computation:

   For $k = 0, ...e-1$, compute $p_k^{-1} \equiv \prod_{\substack{i=0\\i\neq k}}^{e-1}(b_k - b_i)^{-1}$; (If for some $k, i,$ $(b_k - b_i)^{-1}$ does not exist then factor $N$ using $\gcd(b_k - b_i, N)$ and halt);

   $v \equiv \sum_{k=0}^{e-1} b_k^e \cdot p_k^{-1} \bmod N$;
2. Real-time computation: $x \equiv e^{-1} \cdot ((\sum_{k=0}^{e-1} c_k p_k^{-1}) - v) \bmod N$.

The complexity of the pre-computation is $O(e \log^2(e))$ (see Appendix), and the complexity of the real time computations is $O(e)$.

## 3 Special Case

### 3.1 Finite Differences

We now consider the special case where the $e$ cryptograms are of the form $c_i \equiv (ax + b \cdot i)^e \bmod N,\ i = 0, 1, ...e - 1,$ for any known constants $a, b \in \mathbb{Z}_N,$ where $\gcd(a, N) = \gcd(b, N) = \gcd(e!, N) = 1$. The special linear relations among these cryptograms allows us to deterministically compute $x$ in overall $O(e)$ $\mathbb{Z}_N$-operations. As before $x$ denotes an indeterminate variable.

**Definition 2.** *For $h$ a polynomial over any ring let $\Delta^{(0)}(x) \equiv h(x)$, and let*

$$\Delta^{(i)}(x) \equiv \Delta^{(i-1)}(x + 1) - \Delta^{(i-1)}(x), i = 1, 2, ...$$

It is easy to see that the degree of the polynomials resulting from this simpler process keep decreasing as in the case of divided-differences. More precisely:

**Lemma 2.** *In the special case where $x_i \equiv x + i$, and $\gcd(n!, N) = 1$, $[x_0, x_1, ....x_n] \equiv \Delta^{(n)}(x)/n!$*

A similar relation can be derived when $x_i \equiv ax + ib$, for known constants $a, b$. The next two lemmas are stated for general polynomials $h(x)$, although eventually we use them for $h(x) \equiv x^e \bmod N$. Let $m = \deg(h),$ and $0 \leq k \leq m$. By induction on $k$:

**Lemma 3.** $\Delta^{(k)}(x) \equiv \sum_{i=0}^{k} \binom{k}{i} \cdot h(x + i) \cdot (-1)^{k-i} \bmod N.$

For the algorithm we will need explicit formulas for the two leading terms of $\Delta^{(k)}(x)$. Let $h(x) = \sum_{i=0}^{m} a_i x^i$ and let $T^{(k)}_{a_m, a_{m-1}}(x)$ denote the two leading terms of $\Delta^{(k)}(x)$.

**Lemma 4.** $T^{(k)}_{a_m,a_{m-1}}(x) \equiv \frac{(m-1)!}{(m-k)!}x^{m-k-1}(a_m m(x+k(m-k)/2)+a_{m-1}(m-k))$.

*Proof.* We induct on $k$. The basis step is trivial. We verify one more step that is needed later.

$$T^{(1)}_{a_m,a_{m-1}}(x) \equiv x^{m-2}(a_m m(x+\frac{m-1}{2})+a_{m-1}(m-1)) \tag{5}$$

$\Delta^{(1)}(x) \equiv h(x+1)-h(x)$, whose two leading terms are indeed equal to $T^{(1)}_{a_m,a_{m-1}}(x)$ above. Now assume that the two leading terms of $\Delta^{(k-1)}(x)$ are $T^{(k-1)}_{a_m,a_{m-1}}(x) \equiv \alpha x^{m-k+1}+\beta x^{m-k}$, where $\alpha \equiv \frac{(m-1)!}{(m-k)!}a_m m$, and $\beta \equiv \frac{(m-1)!}{(m-k)!}[a_m mk(m-k)/2+a_{m-1}(m-k)]$. The proof can be completed by showing that $T^{(1)}_{\alpha,\beta}(x) \equiv T^{(k)}_{a_m,a_{m-1}}(x)$. This can be done by computing the first difference of $T^{(k-1)}_{a_m,a_{m-1}}(x)$, substituting $\alpha$ for $a_m$ and $\beta$ for $a_{m-1}$ in equation (5) to get the claim.

### 3.2 Related-messages attack with lowered complexity

Using the results of section 3.1 we consider the special case where $x_i \equiv x+i$ (or likewise $x_i \equiv ax+bi$, for known $a,b$) and use the simpler finite-differences to yield overall complexity $O(e)$.

In lemmas 3 and 4 let $h(x) \equiv x^e \bmod N$, where $e \geq 3$. Thus $a_n \equiv 1, a_{n-1} \equiv 0$, and $T^{(e-1)}_{1,0} \equiv e!(x+(e-1)/2)(\bmod N)$. Lemmas 1 and 2 imply that after the $e-1$ finite difference we have a linear congruence $ux+v \equiv w$. Then lemma 4 gives us the values of $u$ and $v$, and lemma 3 tells us how to compute $w$ given the $e$ cryptograms.

Specifically $u \equiv e!$, $v = e!(e-1)/2$ and $w \equiv \sum_{i=0}^{e-1}\binom{e-1}{i}\cdot c_i \cdot (-1)^{e-1+i}$ where $c_i \equiv (x+i)^e$ (all the congruences are taken $\bmod N$). This equation is solvable iff $e!^{-1} \bmod N$ exists, which holds for practical (small) values of $e$. The computation of $w$ dominates, and takes $O(e)$ operations in $\mathbb{Z}_N$ (since $\binom{e-1}{i}$ can be computed from $\binom{e-1}{i-1}$ using one multiplication and one division).

If $x_i \equiv ax+bi \bmod N$, $i=0,1,2...e-1$, for known $a$ and $b$, with $\gcd(a,N)=\gcd(b,N)=1$, we can likewise compute $x$. Given cryptogram $c_i \equiv (ax+b\cdot i)^e \bmod N$ we can transform it into $c'_i \equiv c_i \cdot b^{-e} \equiv (z+i)^e \bmod N$, where $z \equiv xab^{-1} \bmod N$. So

$$x \equiv a^{-1}b[b^e e!)^{-1}\sum_{i=0}^{e-1}\binom{e-1}{i}\cdot c_i \cdot (-1)^{e-1+i} - \frac{e-1}{2}] \mod N.$$

which is computable in $O(e)$ $\mathbb{Z}_N$ operations.

## 4 Conclusions

We have shown new attacks on RSA-encryption assuming known explicit linear relations between the messages. Our attacks require more information (i.e., intercepting more cryptograms), but they run faster than previously published attacks. In some practical cases they run three orders of magnitudes faster than previous attacks. This should be taken into consideration when designing very compact cryptosystems (e.g., for RFID tags), although the default should be using some form of protection like OAEP+ to destroy such known relations. Our attack tools are borrowed from numerical analysis and adapted to handle formal polynomials defined over finite rings.

Open problems: Can these or similar tools be used to attack other cases of known relations, such as implicit linear relations or explicit non-linear relations?

## Acknowledgements

Special thanks go to Gideon Yuval who suggested looking into divided differences, and to Peter Montgomery who made numerous valuable suggestions and corrections. We also thank Don Coppersmith, Kamal Jain, Adi Shamir, and Venkie (Ramarathnam Venkatesan), for helpful discussions on earlier applications of the finite difference technique. Finally, we thank PKC'05 reviewers who made valuable suggestions that improved this paper.

## References

1. Aho Hopcroft and Ullman: "The Design and Analysis of Computer Algorithms", Addison Wesley, 1974, ISBN 0-201-00029-6.
2. D. Boneh: "Twenty Years of Attacks on the RSA Cryptosystem", in Notices of the American Mathematical Society (AMS), Vol. 46, No. 2, pp. 203–213, 1999.
3. M. Bellare and P. Rogaway: "Optimal asymmetric encryption", Eurocrypt'94: 92-111.
4. Don Coppersmith, Matthew Franklin, Jacques Patarin, Michael Reiter: "Low-Exponent RSA with related Messages", Proc. of Eurocrypt'96, LNCS 1070, pp. 1-9.
5. E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern: "RSA-OAEP Is Secure Under the RSA Assumption", J. Crypt. Vo. 17, No.2, March'04, pp. 81-104 (Springer Verlag).
6. Ronald Rivest, Adi Shamir, Leonard M. Adleman: "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", CACM 21(2): 120-126 (1978).
7. Volkov, E.A., "Numerical Methods". New York: Hemisphere Publishing Corporation, pp.48, 1987.
8. Whittaker, E. T. and Robinson, "The Calculus of Observations: A Treatise on Numerical Mathematics", 4th ed. New York: Dover, pp. 20-24, 1967.

## Appendix: The Complexity of the Pre-Processing

The following algorithm, due to Peter Montgomery, computes the pre-processing phase of Algorithm 1 in $O(e \log^2 e)$ time. We currently do not know of a better algorithm for the general case.

For $k = 0, ...e-1$, we need to compute $p_k = \pi'_k(y) \equiv \prod_{\substack{i=0 \\ i \neq k}}^{e-1} (b_k - b_i)$. We use the observation stated before Formula (1). The algorithm proceeds as follows (time complexity for each step is included in the brackets):

1. Expand the formal polynomial $\pi(y) \equiv \prod_{i=0}^{e-1} (y - x_i)$ in indeterminate variable $y$ ($O(e \log^2 e)$,as explained below).
2. Compute the formal derivative of $\pi(y)$ ($O(e)$).
3. Simultaneously evaluate the value of the derivative in the given points $b_i,\ i = 0, 1, ...e-1$ ($O(e \log^2 e)$, see [1] pp. 294, Corollary 2).

Expanding step (1) above:

Suppose we have a polynomial multiplication algorithm that works in time $O(n \log n)$, where $n$ is the degree of the polynomials. Multiply pairs (there are $n/2$ many pairs). Then multiply the resulting $n/4$ pairs at cost $O(2 \log 2)$ each. And so on. There are $\log e$ many levels. Let $e = 2^k$. The total cost is $e \sum_{i=0}^{k} i = O(e \log^2 e)$.

Note that if the $b_i$ happen to be some powers of one primitive $n_{th}$ root of unity, $w \in Z_N$, then we can use DFT in $O(n \log n)$. However, for arbitrary $b'_i s$ chances to have this condition with $n = O(e)$ are negligible.