Batch Diffie-Hellman Key Agreement Systems

Yacov Yacobi\textsuperscript{1} \quad Michael J. Beller\textsuperscript{2}

October 1995\textsuperscript{3}

Abstract.
Batch Diffie-Hellman key agreement schemes are described, motivated, and analyzed for security and efficiency.

1 Introduction

RSA (Rivest, Shamir and Adleman) \cite{8} is today’s most popular public key encryption scheme. Batch-RSA \cite{4} is a method to compute many \((n/log^2(n))\), where \(n\) is the security parameter, throughout logarithms are to the base 2) RSA decryption operations at a computational cost approaching that of one normal decryption. It requires that all the operations use the same modulus, but distinct, relatively prime in pairs, short, public exponents. A star-like key agreement scheme could use such a system to slash complexity at the center. We show a real life example of such a system – secure portable telephony. Unfortunately, in this system Batch-RSA can not be employed effectively, due to a delay component which arises from the nature of RSA key exchange. We then show that mathematical ideas similar to Fiat’s can lead to a Batch-Diffie-Hellman key agreement scheme, that does not suffer such delay and is comparable in efficiency to Batch-RSA. We prove that this system is as hard to break on the average as RSA with short public exponent, where the average is over all messages in the maximal cyclic subgroup of \(\mathbb{Z}_N^\times\), \(N\) being the modulus.

The current system may also have application in Discrete-log based e-cash systems, where a bank can process many e-coins in parallel, and in DL based key escrow systems. The gain factor of \(n/log^2(n)\) is becoming more significant as we move from \(n = 512\) to \(n = 1024\) and beyond.

Section 2 describes Fiat’s original Batch-RSA method, in section 3 we show how the same ideas could be used with Diffie-Hellman-like system, and we motivate this transition. Section 4 presents security analysis for the newly proposed system.

2 Batch-RSA

Let \(N = p \cdot q\), \(p - 1 = 2p', q - 1 = 2q'\), \(p, q, p', q'\) are primes. Suppose we have to compute \(m_i^{1/e_i} \mod N\), for \(i = 0, 1, 2, \ldots, b - 1\), where \(b\) is the batch size, and the \(e_i\)’s are relatively small (and the inverses \(1/e_i \mod \lambda(N)\) are large, \(\lambda(N) = (p - 1) \cdot (q - 1)/2\).

The main idea of Batch-RSA is to compute first \(c = \prod_{i=0}^{b-1} m_i^{p/e_i} \mod N\), where \(p = \prod_{i=0}^{b-1} e_i\), using a special efficient binary tree structure, to be described later. The second phase is to compute \(m \equiv c^{1/p} \mod N\), which is a full size modular exponentiation (but the cost is spread over \(b\) computations). The last phase is to break \(m \equiv \prod_{i=0}^{b-1} m_i^{1/e_i} \mod N\) into its \(b\) separate

\textsuperscript{1}Microsoft Corp., One Microsoft Way, Redmond, WA 98052
\textsuperscript{2}PenVision Information Systems, Inc. 670 North Beers Rd., Bldg 2 Holmdel, NJ 07733
\textsuperscript{3}Work done at Bellcore, NJ, U.S.A.
components \(m_i^{1/e_i} \mod N\), \(i = 0, 1, 2, \ldots, b - 1\), which is the desired output of our computation. This is done using the binary tree developed in the first step in a very efficient way.

**The binary tree:** To simplify explanations we assume that \(b = 2^k\), for some \(k\). Create a complete binary tree where the leaves are labeled \(m_0, m_1, m_2, \ldots, m_{b-1}\). A path is identified with the corresponding binary sequence \(\eta \in \Sigma^*, \Sigma = \{0, 1\}\). We use the symbol \(\epsilon\) to denote the string of length zero. Right sons are associated with 1, and left sons with 0. We refer to any arc in the tree using the unique path leading to it, i.e., \(\eta \in \Sigma^k\), \(k' < k\) is the arc in depth \(k'\) from the root, which is approached when traversing the tree from the root according to \(\eta\). If \(\eta\) is of length \(k\) then it leads to a leaf labeled \(m_i\) s.t. \(\eta\) is the binary representation of \(i\). Let \(x \in \Sigma\). \(\eta x\) denotes a sequence composed of \(x\) concatenated to the right of \(\eta\). \(\tilde{x}\) denotes the complement of \(x\). Each arc \(\eta\) has label \(l(\eta)\). Arcs in the tree are labeled bottom-up according to the following rule:

- Let \(\eta x\) be a path leading to a leaf associated with message \(m_i\), then \(l(\eta \tilde{x}) = e_i\).
- For \(\eta \in \Sigma^k\), \(k' < k - 1\), \(l(\eta x) = l(\eta \tilde{x}) \cdot l(\eta 0)\).

The above labeling procedure is independent of the actual messages \(\{m_i\}\). It depends on the exponents only, hence if those are fixed, this procedure may be done off-line, once and for all. Nodes in the tree contain data, which depend on both the messages and the exponents. We refer to each node by the path leading to it. The data stored in node \(\eta\) is denoted \(d(\eta)\). Initially the content of each leaf \(i\) is the corresponding message \(m_i\). The content of each node in the tree is computed bottom-up, after its sons were computed using \(d(\eta) = d(\eta 0)^{l(\eta 0)} \cdot d(\eta 1)^{l(\eta 1)} \mod N\).

It follows that the content of the root is the desired \(d(e) = \prod_{i=0}^{b-1} m_i^{e_i} \mod N\). This concludes the first phase.

After computing \(M \equiv d(e)^{1/p} = \prod_{i=0}^{b-1} m_i^{1/e_i} \mod N\) (second phase), we use the tree (top-down) to break \(M\) into its components (third phase). This is done recursively as follows.

Let \(0, 1, \ldots, q - 1\) be the leaves associated with the left son of the root (and \(q, q + 1, \ldots, b - 1\) with the right son). Note that \(l(0) = \prod_{i=0}^{q-1} e_i\), \(l(1) = \prod_{i=q}^{b-1} e_i\), and \(p = l(0) \cdot l(1)\). Using the Chinese Remaindering algorithm [1] compute \(X\), s.t. \(X \equiv 0 \mod e_i\), \(i = 0, \ldots, q - 1\), and \(X \equiv 1 \mod e_i\), \(i = q, \ldots, b - 1\). Here we use the fact that the \(e_i\)’s are relatively prime in pairs. From the construction of \(X\) it follows that \(X \equiv 0 \mod l(0)\) and \(X \equiv 1 \mod l(1)\), hence there exist \(X_0\) and \(X_1\) s.t. \(X = l(1) \cdot X_1 + 1 = l(0) \cdot X_0\). Denote \(M_0 = \prod_{i=q}^{b-1} m_i^{1/e_i} \mod N\), and \(M_1 = \prod_{i=0}^{q-1} m_i^{1/e_i} \mod N\). For convenience we use the shorthand \(l_0, l_1, d_0, d_1\) instead of \(l(0), l(1), \ldots\), etc. Since \(M_0 \equiv d_0^{l_0/p} \mod N\), and \(M_1 \equiv d_1^{l_1/p} \mod N\), we have \(d_1 \equiv M_1^{l_0} \mod N\) and \(d_0 \equiv M_0^{l_1} \mod N\). The reader can easily verify that \(M_0 \equiv M^{X_0} / (d_0^{X_1} \cdot d_1^{X_0}) \mod N\), and this is the computation carried in the third phase when going down from the root to its left son. The right son is simply \(M_1 \equiv M / M_0 \mod N\). The \(d’s\) were computed in phase I, and are stored in the nodes. The \(X’s\) do not depend on the messages and may be computed off-line. The process repeats recursively, until at the leaves the desired output is reached.

As is shown in [4] the total complexity of a batch computation approaches that of one full size exponentiation, so the gain factor is \(O(n/\log^2 n)\), which is true for Batch Diffie-Hellman as well.

3 **Batch Diffie-Hellman and Its Motivation**

We consider a star-like graph representing, for example, a personal communications system. In
this graph, the center, through the leaves (called “ports”) agrees upon a session key with each of the nomadic units (called “portables”). The ports are part of the network, and are trusted. Each port holds a secret key that is used to authenticate the network in a transaction. As is shown at the end of the section on security, a port has sufficient data to factor the modulus. This implies that for RSA based schemes, each port must be trusted at the highest level. In the case of DL based systems, we can still have large enough factors $p$ and $q$ such that DL modulo each of them is difficult (i.e. even after factorization). In practice ports sometimes have poor physical protection, in which case it makes sense to do crypto-calculations on behalf of ports in some better protected central office. The port in this case is reduced to just an antenna+transmitter.

A naive system might assign a fixed public exponent $e_i$ to port $i$. However, the center, when batch processing takes place, can not know ahead of time which ports will be active (assuming, for example, that a portable communicates via the nearest port, and only when it requires some service from the center). Thus the use of this fixed assignment will require an untenable pre-processing phase (since pre-computed binary trees will be required for all possible combinations of ports). Instead, we can precompute a tree for a group of exponents associated with the center (not the ports) and assign these exponents to individual key-agreement transactions dynamically, upon service request by a portable.

A representative RSA-based key agreement protocol is one where each of the parties picks a random number, encrypts it with the counterpart’s public key, and sends it to the counterpart. The parties decrypt the received cryptograms, and use some combination of the two random numbers as a session key. Given the constraint of dynamic exponent assignment (described in the previous paragraph), it is not possible for a portable to obtain the center’s public key (which for RSA includes modulus and exponent) until after the portable has requested service, because this is when the “batch” would be defined and exponents assigned. Thus, a batch implementation of this protocol would allow batch processing to begin only after the last encrypted message is received at the center. This would mean that batch processing would incur a delay which is dependent upon the response time of the slowest portable in the batch. (For example, a noisy radio link to a single portable could degrade the key-agreement response times of the network to other portables.) This property is most likely unacceptable in any real-time system (e.g. the users are waiting to make calls). While this delay is asymptotically immaterial (a constant), in practice it is significant.

A Diffie-Hellman key agreement scheme with comparable security does not suffer from such a delay. When a portable signals its request for key agreement, it can accompany the request with sufficient data for the center to begin batch processing.

The current system may also have application in Discrete-log based e-cash systems, where a bank can process many e-coins in parallel, and in DL based key escrow systems. The gain factor of $n/log^2(n)$ is becoming now more significant as we move from $n = 512$ to $n = 1024$ and beyond.

The following is a basic DH scheme, as was first published in [3]. Let $(S_i, P_i)$ be the secret and public keys, respectively, of portable $i$, $i = 1, \ldots, n$, and let $(S_j', P_j')$ be the secret and public key dynamically assigned to port $j$. The central facility is trusted by all ports. For simplicity we describe here Batch-DH for the simplest DH system.

In the basic Diffie-Hellman scheme there is some prime modulus, $N$, common to the whole system, and some primitive element of $GF(N)$, denoted $\alpha$, s.t. $(\forall i)P_i \equiv \alpha^{S_i} \mod N$, similarly, $(\forall j)P_j' \equiv \alpha^{S_j'} \mod N$, and a session key between $i$ and $j$ is $SK_{ij} \equiv \alpha^{S_i - S_j'} \mod N$, efficiently computable exclusively by $i$ and $j$ (or the center on behalf of $j$). Many other variations exist. For example, sometimes it may be desirable to choose a composite modulus (in which we refer to the
system as CDH) \[9\] \[7\], and most of the time we need key agreement systems that authenticate the users, and are dynamic \[10\].

In order for the central authority to be able to use Batch-DH we have to introduce additional constraints.

- First, the modulus should be a composite, with secret prime factorization (two large primes) known only to the trusted ports (or trusted central facility if it does cryptographic computations on behalf of ports), and such that factorization of $N$ is hard. The base element is created as described in Lemma 1 (see section on security).

- Second, the secret key of each port $j$ (or the $j$’th member of a group of secret keys available for dynamic assignment at the center), $S_j^0$, is not chosen at random. Rather a relatively small $e_j$, is chosen, and its multiplicative inverse modulo $(N)$, is computed. This is $S_j^0$. As before, $P_j^0 \equiv \alpha^{S_j^0} \mod N$. For modulus of size $n$ bits we need $e_j < \log_2(n)$.

- Third, the $e_j$s must be relatively prime in pairs.

We remind the reader that portables are untrusted, while ports are trusted (or else their crypto computations are done in a safe trusted central facility). Short public exponents and the corresponding long secret exponents are given only to trusted parties.

4 Security

The following lemma is well known. As before, let $p - 1 = 2p'$, and $q - 1 = 2q'$, where $p'$ and $q'$ are large primes.

**Lemma 1:** Let $\alpha$ and $\beta$ be generators of the multiplicative groups $Z_p^*$ and $Z_q^*$, respectively, and let $\gamma \in Z_N^*$, $\gamma \equiv \alpha \mod p$, $\gamma \equiv \beta \mod q$, (i.e. $\gamma$ is obtained from $\alpha$ and $\beta$ using Chinese Remaindering). Then $\gamma$ generates a maximal cyclic subgroup of $Z_N^*$, it is of size $\lambda(N) = (p - 1)(q - 1)/2$ (for our choice of $p$ and $q$). We denote this cyclic subgroup $M$.

**Comment:** In fact, for the above choice of $p$ and $q$, if we choose $\gamma$ at random with uniform distribution over $Z_N^*$, we get a generator for $M$ with probability $\approx 1/8$: If we pick $\gamma$ with uniform distribution in $Z_N^*$, then with probability $1/2$ it is in $M$, with probability $\approx 1/2$ we get $\text{ord}_p(\gamma) = p - 1$, and $\text{ord}_q(\gamma) = q - 1$ (there are $\phi(q - 1)$ different generators of $Z_q^*$, see e.g. [6], Proposition II.1.2, so for $q - 1 = 2q'$ we have $\phi(q - 1) = q' - 1$, which means about half the elements are generators).

We now give evidence to the security of our scheme, namely, if we assume that RSA with short public key is hard to break on the average, over the subset $M$, then so is restricted CDH. This follows from Lemma 2, and the corollary which immediately follows it. RSA is usually used with short public exponents. If such RSA was easily breakable over $M$, then it would not be considered secure, since $M$ covers about half the instances.

Lemma 2 says that Composite Diffie-Hellman key agreement scheme is at least as hard to break on the average as RSA for messages in $M$, to within a small constant.

The basic problems are:

**The RSA cracking problem :**

Input: $(N, e, c)$ s.t. $e$ is invertible mod $\lambda(N)$ and $c$ is invertible mod $N$.

Output: $m$ between 1 and $N - 1$ s.t. $c \equiv m^e \mod N$.

The solution is uniquely defined by $m \equiv c^{\phi^{-1}} \mod N$ with $x \equiv e^{-1} \mod \lambda(N)$. 

The DH-cracking problem is:
Input: \(N,a,b,c\) s.t. \(b\) and \(c\) are powers of \(a\) mod \(N\). Output: \(a^{xy} \mod N\) s.t. \(b = a^x \mod N\) and \(c = a^y \mod N\) are uniquely defined mod \(\text{order}_N(a)\), and so is \(xy\). Thus, the solution is uniquely defined.

When \(N\) is a composite the DH system is called Composite DH (CDH). A CDH system is restricted if \(a\) (and hence \(b, c\)) are restricted to the multiplicative subgroup \(M\), as defined above. Similarly, RSA is restricted if the message, \(m\) and hence the cryptogram, \(c\), are in \(M\).

**Lemma 2:** Suppose there exists an algorithm that breaks restricted Composite Die-Hellman key agreement scheme, with modulus \(N = pq\), \(p - 1 = 2p'\), \(q - 1 = 2q'\), where \(p'\) and \(q'\) are primes, in time \(t(n)\), with probability \(q(n)\), where \(n = \log_2(N)\) is a security parameter. Then there exists an algorithm to break restricted RSA in time \(t(n)\) with probability \(q(n)/8\).

**Proof:** Assume there exists an oracle \(AL\), s.t. for all \(N\) and for all \(A, B, C \in M\) \(AL(N, A, B, C) = A^{xy} \mod N\), where \(B = A^x \mod N\), and \(C = A^y \mod N\), and let \(e \equiv x^{-1} \mod \lambda(N)\).

Given a restricted RSA cracking problem, we use oracle \(AL\) to solve it, as follows:

1. Pick a random \(\gamma \in Z_N^*\) (with probability 1/8 it is a generator of \(M\), see comment after Lemma 1).
2. Compute \(\beta \equiv \gamma^e \mod N\).
3. Call oracle \(AL(N, \beta, \gamma, c)\), where \(c\) is the cryptogram given in the RSA problem.

Suppose \(\gamma\) which was picked in the first step is a generator of \(M\). Then the oracle’s answer in the last step is \(m\), because \(m \in M\) and hence \((\exists y)[m \equiv \gamma^y]\), hence \(\gamma \equiv \beta^x, c \equiv \beta^y, m \equiv \beta^{xy}\), where all these congruences are modulo \(N\).

Once \(\gamma\) is fixed, the above mapping is one to one, therefore, the reduction is measure preserving \([2]\), and therefore preserves average case complexity. Success probability of this reduction is \(q(n)/8\) from step 1.

Q.E.D.

**Corollary:** The reduction of Lemma 2 does not depend on the length of \(e\). The case in which we are interested is CDH with short inverse. RSA with short public key reduces to CDH with exactly one short inverse by the same construction.

**Lemma 3:** Composite Diffie-Hellman key agreement scheme, in which exactly one of the exponents has a short inverse (< \(O(\log(N))\)) is at most as hard to break on the average (over all messages) as RSA with short public key.

**Proof:** Let \(AL2\) be an oracle that solves RSA problem, where the public exponent \(e\) is short \((e < O(\log(N)))\), i.e. \(AL2(N, e, c) = m, \text{s.t. } m^e \equiv c \mod N\). Given a CDH problem defined by Input: \(N, a, a^x, a^y\); Output: \(a^{xy} \mod N\), where \(x^{-1} \equiv e \mod \lambda(N)\) is short, we use oracle \(AL2\) to solve it as follows:

1. Find \(e, \text{s.t. } (a^x)^e \equiv a \mod N\) (since \(e\) is logarithmic exhaustive search can be done in polynomial time).
2. Call oracle \(AL2(N, e, a^y)\).

The answer of oracle \(AL2\) is \(a^{xy}\), as required.
In this reduction the mapping is one to one. Therefore, it is measure preserving [2], and therefore preserves average case complexity.

Q.E.D.

The following lemma shows that a (trusted) port (or whoever does the cryptographic computations on behalf of a port) has sufficient data to factor the modulus.

**Lemma 4:** Suppose $N = pq$, $p$ and $q$ are large primes, such that $p - 1 = 2p'$, and $q - 1 = 2q'$, where $p'$ and $q'$ are primes, $n = \log_2(N)$, and $xy \equiv 1 \mod \lambda(N)$, s.t. $x$ is of size $\log(n)$ bits. Then there exists a polynomial time algorithm (in $n$) that given $y$ factors $N$.

**Proof:** Try all candidates $x$ of size $\leq \log(n)$ bits. For each compute $xy - 1$. This is a small multiple of $\lambda(N)$ for the right $x$, so one can find all small factors of $xy - 1$, the rest is $p'q'$, which yields $N$’s factorization.

Q.E.D.

The next lemma states that if we use a CDH where both parties have secret exponents with short inverses, then a complete outsider (as opposed to insider as in lemma 4) can effectively calculate the session key.

**Lemma 5:** Composite Diffie-Hellman key agreement scheme, in which both exponents have short inverses is easily breakable for $6/\pi^2$ of the instances.

**Proof:** If both secret exponents $(x, y)$ have short inverses $(e_i, e_j)$ then, as before, the adversary can find them by exhaustive search (raising the public keys to the power of $e...$ until he gets $\alpha$). If in addition the two short inverses are relatively prime (happens with probability $6/\pi^2$, see a theorem by Dirichlet in Knuth II, pp. 324) the adversary can break the system. First he finds $a$ and $b$ such that $ae_i + be_j = 1$, using extended Euclid’s gcd algorithm [1]. Multiplying this equation by $xy$ we get $a y + b x \equiv xy \mod \lambda(N)$. It follows that the adversary can compute the session key $\alpha^{xy} \equiv (\alpha^x)^b (\alpha^y)^a \mod N$.

Q.E.D.

Of course, the reduction of Lemma 2 does not hold when both exponents have short inverses (we remind the reader that portables use secret exponents with long inverses).

5 Acknowledgment

We wish to thank Rafi Heiman and Arjen K. Lenstra for reviewing earlier versions of this paper, and for their many helpful comments. Special thanks go to anonymous referee of J. Crypt. for most educating discussions, which greatly improved this paper.

References


