Secure Trapdoor Hash Functions Based on Public-Key Cryptosystems

Gary R. Greenfield

Sarah Agnes Spence

Secure Trapdoor Hash Functions Based on Public-Key Cryptosystems

Gary R. Greenfield and Sarah A. Spence
Department of Mathematics and Computer Science
University of Richmond
Richmond, Virginia 23173

December, 1995

E-mail: grg5l@mathcs.urich.edu, sas0m@mathcs.urich.edu

TR-95-02

1 Introduction.

In cryptology, the study of *message digest algorithms* leads naturally to the study of *secure hash algorithms.* For background and motivation on these topics the reader is urged to consult [7] [9] or [4]. By a *hash* or *compression* algorithm we mean a function h such that for a message M of length $|M|$, $|h(M)| < |M|$. Usually M is represented as a bit-string. More formally we have:

DEFINITION 1.1 A hash algorithm is a (partial) function $h : Z_2^k \longrightarrow Z_2^l$ where $l < k$.

A hash function is *secure* if it is computationally infeasible to find *collisions.* There are a variety and hierarchy of collision problems one can consider. Of primary importance to us are the following three collision problems:

Type I. Find $M \neq M'$ such that $h(M) = h(M')$.

Type II. Given M and $h(M)$, find $M \neq M'$ such that $h(M) = h(M')$.

Type III. Given c, find $M \neq M'$ such that $h(M) = c = h(M')$.

It is apparent that exhibiting a solution to a Type II collision problem also furnishes a solution to a Type I collision problem.

By a *trapdoor* hash function we mean a hash function that has an intrinsic weakness known only to the designer of the hash function, a weakness which allows he or she to find collisions that would be concealed from, and presumably difficult to discover by, other "expert" designers or "attackers" analyzing the system.

In this paper we systematically consider examples representative of the various families of public-key cryptosystems to see if it would be possible to incorporate them into trapdoor hash functions, and we attempt to evaluate the resulting strengths and weaknesses of the functions we are able to construct. We are motivated by the following question:

QUESTION 1.2 How likely is it that the discoverer of a heretofore unknown public-key cryptosystem could subvert it for use in a plausible secure trapdoor hash algorithm?

In subsequent sections, our investigations will lead to a variety of constructions and bring to light the non-adaptability of public-key cryptosystems that are of a "low density." More importantly, we will be led to consider from a new point of view the effects of the unsigned addition, shift, exclusive-or and other logical bit string operators that are presently used in constructing secure hash algorithms: We will show how the use of public-key cryptosystems leads to "fragile" secure hash algorithms, and we will argue that circular shift operators are largely responsible for the security of modern high-speed secure hash algorithms.

2 RSA and Proof of Concept

In this section, we document the first crude example that inspired our subsequent research on this topic. We begin with a brief review of the most well-known public-key cryptosystem, the RSA cryptosystem, a member of the family of algorithms based on modular exponentiation.

Let p and q be odd primes, and set $n = pq$. Choose e such that $(e, \varphi(n)) = 1$, where φ is the Euler φ-function, and use the Euclidean Algorithm to solve

$$ed \equiv 1 \pmod{\varphi(n)}.$$

Publish e and n as the public key, and reserve p, q and d as the private key.[1] Encrypt a message M, where $0 < M < n$, using encryption function E given by

$$E(M) = M^e \pmod{n},$$

and decrypt ciphertext C using decryption function D given by

$$D(C) = C^d \pmod{n}.$$

We are ready for our first example.

Example 2.1 Let e_1 and e_2 be a *pair* of public keys with corresponding private keys d_1 and d_2 for the same RSA modulus n. We assume n is on the order of k bits (*i.e.*, $n \sim 2^k$) so that a message M may be viewed as a bit-string of length k, and we consider

$$h : Z_2 \times Z_2^k \longrightarrow Z_2^k$$

[1] In presenting the mathematical essence of RSA, we omit such implementation issues as the need for large, "safe" primes of fifty to one hundred decimal digits, small public exponents, etc.

defined by

$$h(b, M) = bM^{e_1} + (1-b)M^{e_2} \pmod{n}.$$

To solve the Type II collision problem, given the bit-string $(0, M)$ of length $k+1$ and its hash $h(0, M) = M^{e_2}$, we find

$$h(1, M^{e_2 d_1}) = M^{e_2 d_1 e_1} = (M^{e_1 d_1})^{e_2} = M^{e_2} \pmod{n}.$$

Similarly, given $(1, M)$ and its hash $h(1, M) = M^{e_1}$, we have

$$h(0, M^{e_1 d_2}) = M^{e_1 d_2 e_2} = (M^{e_2 d_2})^{e_1} = M^{e_1} \pmod{n}.$$

Moreover, to solve the Type III collision problem, given c we observe that

$$h(1, c^{d_1}) = c = h(0, c^{d_2}).$$

What is disturbing about this example is the trivial Type I collision

$$h(1, M^{e_2}) = M^{e_1 e_2} = h(0, M^{e_1}),$$

and the transparency of the compression function itself due to the fact that it is just simple modular exponentiation. This is even made more apparent as the equivalent formulation

$$h(b, M) = M^{be_1 + (1-b)e_2} \pmod{n}$$

more clearly reveals how exponent selection occurs.

Since the compression achieved in our first example is merely a one bit compression, it is easy understand why we regard this as "proof of concept." To improve upon it, we must find some way to package it in a more classical style, one that incorporates the canonical operators found in message digest algorithms such as the exclusive-or operator, which we denote by $\oplus$, and the bitwise logical-or operator, which we denote by $|$.

Example 2.2 Let k, n, e_1, e_2, d_1, d_2 be as in Example 2.1 above, and let s and t be fixed but arbitrary (invertible) elements in Z_n. Consider the $(2k+1)$-bit to k-bit compression function

$$h : Z_2 \times Z_2^k \times Z_2^k \longrightarrow Z_2^k$$

given by

$$h(b, M_1, M_2) = (M_1^{e_1 s} \oplus M_2^{e_2 s}) \mid (bM_1^t + (1-b)M_2^{e_2 d_1 t}) \pmod{n}.$$

Evidently, we may write

$$h(b, M_1, M_2) = f(M_1, M_2) \mid g(b, M_1, M_2) \pmod{n}.$$

Since for any M_1, M_2,

$$f(M_2^{d_1 e_2}, M_1^{d_2 e_1}) = M_2^{d_1 e_2 e_1 s} \oplus M_1^{d_2 e_1 e_2 s} = M_2^{e_2 s} \oplus M_1^{e_1 s} = f(M_1, M_2),$$

we will have found as a solution to the Type II collision problem

$$h(1-b, M_2^{d_1 e_2}, M_1^{d_2 e_1}) = h(b, M_1, M_2)$$

provided we can verify

$$g(1-b, M_2^{d_1 e_2}, M_1^{d_2 e_1}) = g(b, M_1, M_2).$$

Case 1. If $b = 1$, then

$$g(1, M_1, M_2) = M_1^t \pmod{n},$$

and

$$g(0, M_2^{d_1 e_2}, M_1^{d_2 e_1}) = (M_1^{d_2 e_1})^{e_2 d_1 t} = M_1^t \pmod{n}$$

as desired.

Case 2. If $b = 0$, then

$$g(0, M_1, M_2) = M_2^{e_2 d_1 t} \pmod{n},$$

while

$$g(1, M_2^{d_1 e_2}, M_1^{d_2 e_1}) = M_2^{d_1 e_2 t} \pmod{n},$$

and the verification is complete.

We wish to remind the reader that the use of the exclusive-or operator was for convenience and other commutative binary operators such as ordinary unsigned addition or multiplication in Z_n would serve just as well.

If we overlook the trivial solution to the Type I collision problem

$$h(0,0,0) = 0 = h(1,0,0),$$

the following "tests" that check for obvious collisions provide some evidence to bolster the assertion that the previous example *appears* to be more resistant to a Type I attack.

$$\begin{aligned} h(0,M,M) &= (M^{e_1 s} \oplus M^{e_2 s}) \mid M^{e_2 d_1 t} \pmod{n} \\ h(1,M,M) &= (M^{e_1 s} \oplus M^{e_2 s}) \mid M^{t} \pmod{n}, \end{aligned}$$

$$\begin{aligned} h(0,0,M) &= M^{e_2 s} \mid M^{e_2 d_1 t} \pmod{n} \\ h(1,0,M) &= M^{e_2 s} \mid 0 = M^{e_2 s} \pmod{n} \\ h(0,M,0) &= M^{e_1 s} \mid 0 = M^{e_1 s} \pmod{n} \\ h(1,M,0) &= M^{e_1 s} \mid M^{t} \pmod{n}, \end{aligned}$$

$$\begin{aligned} h(0,M_1,M_2) &= (M_1^{e_1 s} \oplus M_2^{e_2 s}) \mid M_2^{e_2 d_1 t} \pmod{n} \\ h(1,M_1,M_2) &= (M_1^{e_1 s} \oplus M_2^{e_2 s}) \mid M_1^{t} \pmod{n} \\ h(0,M_2,M_1) &= (M_2^{e_1 s} \oplus M_1^{e_2 s}) \mid M_1^{e_2 d_1 t} \pmod{n} \\ h(1,M_2,M_1) &= (M_2^{e_1 s} \oplus M_1^{e_2 s}) \mid M_2^{t} \pmod{n}. \end{aligned}$$

The question that arises when looking at the pervasive use of exponents in this system, *all* related to the same RSA modulus n, is whether any "information leakage" might occur. Specifically, how secure are the exponents e_1, e_2, d_1 and s in view of the fact that $e_1 s, e_2 s$, and $e_2 d_1$ are plainly visible. An attacker may not know how the designer explicitly labels the exponents, but since it is true that $(e_1 s)(e_2 s)^{-1} = e_1 d_2 = (e_2 d_1)^{-1} \pmod{\varphi(n)}$, what assumptions can an attacker make about the exponents, and what can an attacker conclude based on his or her assumptions?

If we now demand that our exponent t be invertible in Z_n, then we can construct a third exponent pair $e_3 = t$ and $d_3 = t^{-1}$, and we can solve the Type III collision problem for our previous example as follows.

Example 2.3 With hypotheses as above, and

$$h : Z_2 \times Z_2^k \times Z_2^k \longrightarrow Z_2^k$$

given by

$$h(b, M_1, M_2) = (M_1^{e_1 s} \oplus M_2^{e_2 s}) \mid (b M_1^{e_3} + (1-b) M_2^{e_2 d_1 e_3}) \pmod{n}$$

we know that h satisfies

$$h(b, M_1, M_2) = h(1-b, M_2^{d_1 e_2}, M_1^{d_2 e_1}).$$

For the Type III collision problem, given c we must find *two* messages that hash to c. We have

$$h(1, c^{d_3}, c^{e_1 d_2 d_3}) = (c^{e_1 d_3 s} \oplus c^{e_1 d_2 d_3 e_2 s}) \mid c^{d_3 e_3} = c$$

while

$$h(0, c^{d_3}, c^{e_1 d_2 d_3}) = (c^{d_3 e_1 s} \oplus c^{d_2 e_1 d_3 e_2 s}) \mid c^{d_2 e_1 d_3 e_2 d_1 e_3} = c.$$

Certainly this also is an unsatisfying, seemingly artificial solution to the Type III collision problem, but since we do not know of any general techniques for solving exponential equations involving the exclusive-or operator, it is the best we can offer.

3 The Knapsack Family and the Significance of Density

After the exponentiation family, the next most widely studied family of public-key cryptosystems are those based on knapsack problems. Even though there is ample evidence in the literature to suggest that knapsack cryptosystems are weak and should be avoided, they are still of considerable theoretical interest. It was not possible to construct some version of a secure trapdoor hashing scheme for every knapsack cryptosystem we considered. When we examined the two most popular knapsack examples, the original Merkle-Hellman Knapsack Cryptosystem and the Graham-Shamir Knapsack, we concluded that they probably could not be incorporated into trapdoor hash functions at all. To pinpoint the reasons for this let us consider the Merkle-Hellman Knapsack in more detail.

Recall that a super-increasing knapsack S is a set $\{x_1, ..., x_k\}$ of positive integers, the knapsack *vectors*, which satisfy $2x_i < x_{i+1}$ for all $i < k$. Given such a knapsack, there is an efficient algorithm for finding the binary coefficients ε_i in any linear combination of the form $x = \sum \varepsilon_i x_i$. For Merkle-Hellman, we choose u such that $2x_k < u$ and w relatively prime to u so that we can form *public* instances $wS = \{x_1', ..., x_k'\}$ of S. If we let $< X, Y >_u$

denote $X \cdot Y \pmod{u}$, then the encryption function associated to wS is $E : Z_2^k \longrightarrow Z_u$ described by the equation

$$E(M) =< M, wS >_u .$$

The decryption algorithm is just the algorithm for recovering coefficients applied to the linear combination $w^{-1}E(M) =< M, S >_u$.

Based on our experiences with previous examples, we might expect that a naive attempt to create a secure trapdoor hash, such as

$$h : Z_2^k \times Z_2^k \longrightarrow Z_2^l,$$

given by

$$h(M_1, M_2) =< M_1, w_1 S >_u \oplus < M_2, w_2 S >_u,$$

where $l = \lceil \lg u \rceil$ is the least number of bits required to write an integer in Z_u in binary, would furnish Type II collisions according to the computation:

$$\begin{aligned} h(w_1^{-1}w_2M_2, w_2^{-1}w_1M_1) &= < (w_1^{-1}w_2M_2, w_1S >_u \oplus < w_2^{-1}w_1M_1, w_2S >_u \\ &= < M_2, w_2S >_u \oplus < M_1, w_1S >_u \\ &= h(M_1, M_2). \end{aligned}$$

But closer inspection reveals that there is a flaw, because we have no assurance that $w_1^{-1}w_2M_2$ (respectively $w_2^{-1}w_1M_1$) *is* a message vector since $w_1^{-1}w_2$ (respectively $w_2^{-1}w_1$) is not zero or one. In fact, such a product cannot equal zero, and it equals one provided $w_1 = w_2 \pmod{u}$, which occurs precisely when $w_1 = w_2$, since $0 < w_1, w_2 < u$.

Therefore we see that when working with standard operators such as exclusive-or and unsigned addition applied to knapsack vectors there are two issues that one must consider: the *density* of the knapsack and the *representation* of the message vectors. Specifically, if we try to "decouple" $h(M) = c$ as $c = c_1 \oplus c_2$ then we must verify that c_1 and c_2 are in the image space of h which, for the particular case at hand, means that they are linear combinations arising from message vectors. We remark that in our present context, density of a knapsack algorithm can be precisely defined as the ratio $2^k/u$, the ratio of the possible 2^k knapsack sums $\sum \varepsilon_i v_i$ to u, the size of the "space."

To give an example of a successful trapdoor hash based on knapsacks, we turn to a knapsack system based on complementing sets due to Webb

[10]. Though the system seems neither to be widely known nor to have been analyzed for cryptographic weaknesses, it is of interest to us because it has density one! For the details on the construction of complementing sets we refer the reader to Webb's paper.[2]

EXAMPLE 3.1 Let $A_1, A_2, ..., A_j$ be complementing sets for the positive integer n. If we write $A_i = \{a_{i,0}, ..., a_{i,m_i-1}\}$, then any "message" M, $0 \leq M < n$ is *uniquely* decomposed as $\sum_{i=1}^{j} x_i N_i$ where $0 \leq x_i < m_i$, and fast decryption of the "private" encryption $c_d = \sum a_{i,x_i}$ is possible. Note that x_i is an *index* to an element in the set A_i. The idea now is to "disguise" each A_i to a set G_i so that the decryption of the "public" encryption $c_e = \sum x_i g_{i,x_i}$ is infeasible, but one can (privately) transform c_e to c_d. The construction of G_i takes place in two steps. First, let $F_i = r_1(A_i + t_i) \pmod{u_1}$ where $u_1 > n$, and then let $G_i = r_2 F_i \pmod{u_2}$ where $u_2 > j u_1$. For n large, choosing $u_1 = n + 1$ will not effect the density, but the choice of u_2 reduces the density to $1/j$.

To envision what this system would look like in more concrete terms, we are forced to use a prohibitively small example.

Instance #1.

$n = 12$, $j = 2$.
$A_1 = \{0, 1, 6, 7\}$ so $m_1 = 4$.
$A_2 = \{0, 2, 4\}$ so $m_2 = 3$.
$N_1 = 1$, $N_2 = 4$.
$u_1 = 13$, $r_1 = 3$, $t_1 = 4$, $t_2 = 6$.
$F_1 = \{12, 2, 4, 7\}$.
$F_2 = \{5, 11, 4\}$.
$u_2 = 27$, $r_2 = 7$.
$G_1 = \{3, 14, 1, 22\}$.
$G_2 = \{8, 26, 1\}$.

Instance #2.

$n = 12$, $j = 2$.
$A_1 = \{0, 4, 8, 1, 5, 9\}$ so $m_1 = 6$.
$A_2 = \{0, 2\}$ so $m_2 = 2$.
$N_1 = 1$, $N_2 = 6$.
$u_1 = 13$, $r_1 = 5$, $t_1 = 2$, $t_2 = 3$.

[2] The reader is forewarned that we found it necessary to completely change Webb's notation in order to maintain consistency in our presentation.

$F_1 = \{10, 4, 11, 2, 8, 3\}$.
$F_2 = \{2, 12\}$.
$u_2 = 27$, $r_2 = 4$.
$G_1 = \{13, 16, 17, 8, 5, 12\}$
$G_2 = \{8, 21\}$.

The weakness in this example is easily observed since G_1 and G_2 are not disjoint in each instantiation. To implement the hash, consider

$$h : Z_{12} \times Z_{12} \longrightarrow Z_2^6$$

given by

$$h(M_1, M_2) = E_1(M_1) \oplus E_2(M_2),$$

where E_i invokes the public encryption algorithm using the i-th instantiation. For example, $h(1, 8) = 010110 \oplus 100110 = 110000$, because $E_1(1) = E_1(1 \cdot 1 + 0 \cdot 4) = 14 + 8 = 22$ and $E_2(8) = E_2(2 \cdot 1 + 1 \cdot 6) = 17 + 21 = 38$. There is now a *probabilistic* scheme for searching for collisions: Decouple $h(M_1, M_2) = c_{e1} \oplus c_{e2}$ to $h(M_1, M_2) = c'_{e1} \oplus c'_{e2}$ and transform the components c'_{e1}, c'_{e2} to c'_{d1}, c'_{d2}. Find the corresponding x'_{i1} and x'_{i2} coefficients and then try to verify that their images give c'_{e1} and c'_{e2}.

4 Idempotent Transformations and Fragility

In our quest to consider a wide variety of public-key cryptosytems, we examined knapsack algorithms using polynomials over finite fields, including the Cooper-Patterson public-key cryptosystem [1] and the Chor-Rivest Algorithm [5]. Eventually we were attracted to a knapsack system introduced by Seberry and Pieprzyk [8] which, though it seemed suspect to us regarding its decryption algorithm, led us to reconsider our RSA examples in a more general context.

Example 4.1 As usual, fix an RSA system using modulus n of k bits and public keys e_1, e_2 with respective private keys d_1, d_2. Consider

$$h : Z_2^k \times Z_2^k \times Z_2^k \longrightarrow Z_2^k$$

defined by

$$h(M_1, M_2, M_3) = M_1^{e_1} M_3 \oplus M_2^{e_2}(1 - M_3) \pmod{n}.$$

It is routine to verify the solution to the Type II collision problem

$$h(M_2^{d_1 e_2}, M_1^{e_1, d_2}, 1 - M_3) = h(M_1, M_2, M_3),$$

and the solution to the Type III collision problem

$$h(c^{d_1}, 0, 1) = c = h(0, c^{d_2}, 0).$$

The previous example hinges upon the introduction of an *idempotent* transformation which can be applied to M_3. Formally, for $I : Z_2^k \longrightarrow Z_2^k$ satisfying I^2 is the identity function (*e.g.*, $I(x) = 1 - x \pmod n$ or $I(x) = x^{-1} \pmod n$ for the integer interpretation of bit strings and $I(x) = \overline{x}$ for the boolean interpretation of bit strings), the $3 : 1$ compression function h_I defined in terms of encryption (respectively decryption) functions E_i (respectively D_i) and binary operators $\circ_i$ is given by

$$h_I(M_1, M_2, M_3) = (E_1(M_1) \circ_1 M_3) \circ_2 (E_2(M_2) \circ_1 I(M_3)).$$

For collisions, we find

$$h_I(M_1, M_2, M_3) = h_I(D_1(E_2(M_2)), D_2(E_1(M_1)), I(M_3)),$$

and

$$h_I(D_1(c), 0, 1) = c = h_I(0, D_2(c), I(1)),$$

are solutions to the Type II and Type III problems respectively.

Such generalized constructions begin to suggest that we are discovering "building blocks" for use in secure hash algorithms that are more in tune with the fast, commercial hashes like MD-5 or SHA [7]. However, even a tentative and optimistic comparison reveals that there is one glaring weakness — it is possible to adjust or slightly modify the commercial grade algorithms through the use of additive constants, additional exclusive-or terms and, most importantly, *circular shifting constants.* This observation leads us to conclude that the secure trapdoor hashing components we are considering are *fragile* in the sense that the introduction of any (circular) shift operator destroys their trapdoor features. To further understand this, denote by $S(M)$ a positive integer in the interval $[0, k-1]$ to be used as a shifting "constant." We remark, however, that we do not rule out the possibility that $S(M)$ is an autokey function, meaning that $S(M)$ could depend on M in a mild way such as being a weight function or a parity function. Then, if we denote the left circular shift operator on a bit string x by y positions as $x << y$, a seemingly innocuous circular shifted version of Example 2.1 would be

$$h(b, M) = b(M^{e_1} << S(M)) + (1 - b)M^{e_2} \pmod n.$$

Given (b, M), the Type II collision with $b = 1$ is easily found to be

$$h(1, M) = M^{e_1} << S(M) = h(0, (M^{e_1} << S(M))^{d_2}),$$

but finding the Type II collision for $h(0, M) = M^{e_2}$ requires one to solve

$$\begin{aligned} h(1, M^x << S(y)) &= (M^x << S(y))^{e_1} << S(M^x << S(y) \\ &= M^{e_2} \pmod n \end{aligned}$$

for *both* x and y. Since this appears daunting, there seems to be little hope for solving collision problems using a more realistic circularly shifted variant, such as the following one patterned after Example 4.1 but incorporating shift constants u_1 and u_2:

$$\begin{aligned} h(M_1, M_2, M_3) &= ((M_1^{e_1 s} << u_1) \oplus (M_2^{e_2 s} << u_2)) \mid \\ & \quad (M_1^t M_3 + M_2^{e_2 d_1 t}(1 - M_3) \pmod n). \end{aligned}$$

Of course the problem we are facing is that circular shift compatibility is counter to the "diffusion" and "substitution" goals of classical cryptography. Thus we are led to the following extremely interesting question which we have been unable to resolve.

Question 4.2 Does there exist a public-key cryptosystem with encryption *equation* $E(M)$ together with some shifting constant u for which it is possible to relate $E(M << u)$ to $E(M), u$, or $E(M) << u$?

5 Cellular Automata PKC, Our Best Example

Another public-key cryptosystem whose cryptological significance is also unclear provides interesting possibilities for trapdoor hashing. It is the perhaps slightly misnamed "cellular automata" public-key cryptosystem of Guan [2]. The system requires a carefully constructed boolean vector-valued public encryption function $E : Z_2^k \longrightarrow Z_2^k$ which we will write in terms of its coordinate functions $e_1, ..., e_k$ as $E(x) = (e_1(x), ..., e_k(x))$ where $x = (x_1, ..., x_k)$.

The associated private decryption *algorithm* which we denote by $D(x)$ depends on the order these coordinate functions are considered. The complete details governing the construction are beyond the scope of this paper.

EXAMPLE 5.1 Let $B : Z_2^k \longrightarrow Z_2^k$ be any vector valued boolean function. By using De Morgan's Laws

$$\overline{a+b} = \overline{a} \cdot \overline{b}$$

and

$$\overline{a \cdot b} = \overline{a} + \overline{b},$$

we are able to write

$$\overline{B(x)} = \overline{B}(\overline{x})$$

and thus we are able to construct a trapdoor hash function

$$h : Z_2^k \times Z_2^k \longrightarrow Z_2^k$$

by letting

$$h(M_1, M_2) = \overline{B(E(M_1))} +_2 \overline{B}(\overline{M_2}),$$

where we have used the unsigned binary addition operator $+_2$ (*i.e.* binary addition with carry), though in fact any commutative binary operator such as exclusive-or, logical-or, logical-and, or even unsigned multiplication will work. To see why this formulation satisfies our trapdoor criteria, observe that

$$\begin{aligned} h(D(M_2), \overline{E(M_1)}) &= \overline{B(E(D(M_2)))} +_2 \overline{B}(\overline{E(M_1)}) \\ &= \overline{B(M_2)} +_2 \overline{B(E(M_1))} \\ &= \overline{B}(\overline{M_2}) +_2 \overline{B(E(M_1))}. \end{aligned}$$

Hence we have exhibited a solution to the Type II collision problem.

We contend the construction found in Example 5.1 is our best construction because of its believability. Recall, our scenario is that a public-key system might be inserted into a hash algorithm and go undetected. This could never happen with the modular exponentiation examples which we have presented: They are quite transparent. The resulting fast and efficient hashing system that would result from the construction above, however, masks the deception much better. To exhibit a concrete instantiation, we use $k = 5$ and continue with an example found in Guan [2]. To simplify its description, we write

$$E(x_1, x_2, x_3, x_4, x_5) = (y_1, y_2, y_3, y_4, y_5),$$

and

$$B(y_1, y_2, y_3, y_4, y_5) = (z_1, z_2, z_3, z_4, z_5).$$

Following Guan, we let

$$\begin{aligned}
y_1 &= x_1x_2 + x_5 \\
y_2 &= x_1x_3 + x_4 \\
y_3 &= x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_5 + x_4x_5 + x_2 \\
y_4 &= x_2x_1 + x_2x_5 + x_3 \\
y_5 &= x_1 + x_2.
\end{aligned}$$

For our arbitrary boolean function B, we let

$$\begin{aligned}
z_1 &= y_1y_3 + y_2y_5 \\
z_2 &= y_1y_2 + y_1y_4 \\
z_3 &= y_1 + y_2 + y_3 + y_4y_5 \\
z_4 &= y_1y_2y_3 + y_4 \\
z_5 &= y_3 + y_5.
\end{aligned}$$

This gives $(B \circ E)(x)$ as

$$\begin{aligned}
z_1 &= x_1x_2x_4 + x_1x_3x_5 + x_1x_2 + x_1x_2x_3x_5 + x_2x_3x_5 + x_4x_5 + x_2x_5 + x_1x_3 + x_1x_4 + x_2x_4 \\
z_2 &= x_1x_2x_5 + x_1x_2x_4 + x_1x_3x_5 + x_4x_5 + x_1x_2 + x_1x_2x_3 + x_2x_5 + x_3x_5 \\
z_3 &= x_5 + x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_5 + x_4x_5 + x_2 + x_1x_2 + x_1x_2x_5 + x_2x_5 + x_2x_3 \\
z_4 &= x_4x_5 + x_2x_4x_5 + x_1x_2 + x_2x_5 + x_3 \\
z_5 &= x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_5 + x_4x_5 + x_1.
\end{aligned}$$

Next, we compute $\overline{B(z)} = \overline{B}(\overline{z})$ as follows:

$$\begin{aligned}
\overline{z_1} &= \overline{y_1} \cdot \overline{y_2} + \overline{y_1} \cdot \overline{y_5} + \overline{y_2} \cdot \overline{y_5} + \overline{y_3} \cdot \overline{y_5} \\
\overline{z_2} &= \overline{y_1} + \overline{y_1} \cdot \overline{y_4} + \overline{y_1} \cdot \overline{y_2} + \overline{y_2} \cdot \overline{y_4} \\
\overline{z_3} &= \overline{y_1} \cdot \overline{y_2} \cdot \overline{y_3} \cdot \overline{y_4} + \overline{y_1} \cdot \overline{y_2} \cdot \overline{y_3} \cdot \overline{y_5} \\
\overline{z_4} &= \overline{y_1} \cdot \overline{y_4} + \overline{y_2} \cdot \overline{y_4} + \overline{y_3} \cdot \overline{y_3} \\
\overline{z_5} &= \overline{y_3} \cdot \overline{y_5},
\end{aligned}$$

and therefore finally obtain, $\overline{B}(x) = (w_1, w_2, w_3, w_4, w_5)$ as

$$\begin{aligned}
w_1 &= x_1x_2 + x_1x_5 + x_2x_3 + x_3x_5 \\
w_2 &= x_1 + x_1x_4 + x_1x_2 + x_2x_4 \\
w_3 &= x_1x_2x_3x_4 + x_1x_2x_3x_5 \\
w_4 &= x_1x_4 + x_2x_4 + x_3x_4 \\
w_5 &= x_3x_5.
\end{aligned}$$

The reason for writing the z's and w's in terms of x's is that they are the equations one needs to implement our hash construction *i.e.*, the z's receive the bits of M_1 as x's and the w's receive the bits of $\overline{M_2}$ as x's.

6 Nested Encryption Methods

In this section we shall exploit the potential for trapdoor hashing arising from nested encryption. The first example does not give the full generality for reasons we shall subsequently explain.

EXAMPLE 6.1 Let $E_1, E_2 : Z_2^k \longrightarrow Z_2^k$ be public-key encryption algorithms, and consider

$$h : Z_2^k \times Z_2^k \longrightarrow Z_2^k$$

defined by

$$h(M_1, M_2) = E_1(M_1) \oplus (E_1 \circ E_2)(M_2).$$

We have the solution to the Type II collision problem

$$\begin{aligned}
h(E_2(M_2), D_2(M_1)) &= E_1(E_2(M_2)) \oplus E_1(E_2(D_2(M_1))) \\
&= E_1(E_2(M_2)) \oplus E_1(M_1),
\end{aligned}$$

where D_2 refers to the *algorithm* for decrypting $E_2(x)$. Moreover, given c and any decomposition of c, $c = c_1 \oplus c_2$, we have the solution to Type III collision problem given by

$$\begin{aligned}
h(D_1(c_1), D_2(D_1(c_2))) &= E_1(D_1(c_1))) \oplus (E_1 \circ E_2)(D_2(D_1(c_2))) \\
&= c_1 \oplus c_2 = c.
\end{aligned}$$

The cellular automaton cryptosystem enjoys the property that the composition of vector-valued boolean functions is again a vector-valued boolean function. Therefore, this nesting scheme is particularly significant because of the “form” the required composition $E_1 \circ E_2$ would assume. It would be of interest to perform some computations to compare E_1 with $E_1 \circ E_2$ if, say, $E_1 = A_1 \circ A_2$ and $E_2 = A_3 \circ A_4$ were based on four “s-fold” invertible linear transformations $A_1, A_2, A_3, A_4 : Z_2^k \longrightarrow Z_2^k$ in the sense of Guan [2].

We should also observe that the RSA version of Example 5.2 is

$$h(M_1, M_2) = M_1^{e_1} \oplus M_2^{e_1 e_2} \pmod{n},$$

with solution to the Type II collision problem

$$h(M_2^{e_2}, M_1^{d_2}) = M_2^{e_1 e_2} \oplus M_1^{e_1 e_2 d_2} = M_2^{e_1 e_2} \oplus M_1^{e_1} \pmod{n}.$$

And we should remark once more that from a design viewpoint there is nothing sacrosanct about using exclusive-or operators. Other commuting binary operators would also be acceptable.

The principal reason for focusing on nesting of encryption algorithms is their applicability to a *set* of encryption algorithms that, unlike RSA, are noncommuting *viz.*, $E_i E_j \neq E_j E_i$ for some $i \neq j$.

EXAMPLE 6.2 Consider g instances $E_1, ..., E_g$ of public encryption schemes, and let

$$h : Z_{2^k}^g \longrightarrow Z_{2^k}$$

be defined by

$$h(M_1, M_2, ..., M_g) = E_1(M_1) \oplus (E_1 E_2)(M_2) \oplus ... \oplus (E_1 E_2 ... E_g)(M_g).$$

Then the solution to the Type II collision problem is found using the identity

$$\begin{aligned} &h((E_2...E_g)(M_g), D_2(M_1), D_3(M_2), ..., D_g(M_{g-1})) = \\ &\quad (E_1 E_2 ... E_g)(M_g) \oplus E_1(M_1) \oplus (E_1 E_2)(M_2) \oplus ... \oplus (E_1 ... E_{g-1})(M_{g-1}), \end{aligned}$$

and there is an “easy” solution to the Type III problem obtained by decrypting componentwise

$$c = c_1 \oplus ... \oplus c_g.$$

Note that the succinct way to write h is

$$h(M_1, ..., M_g) = \oplus_{i=1}^{g} \ (E_1 ... E_i)(M_i).$$

7 Matrix Methods

We were dismayed to find that there was no *viable* word-problem public-key cryptosystem that we could lay hands on, and that another widely touted example of a probabilistic public-key cryptosystem (a system where the encryption of a message results in a *set* of ciphertexts from which to choose), the McEliece PKC based on Goppa codes [8], was unsuitable because of the severe expansion that encryption produced. Unexpectedly, the probabilistic system that we found to be adaptable for a trapdoor hash system was a matrix system invented by Varadharajan and Odoni [11]. It is a system that uses both RSA style exponents and randomly chosen elements. We give a brief description.

Following [11], over Z_m we let g divide the *exponent* of the group of $n \times n$ nonsingular upper triangular matrices[3], and choose e, d such that $ed \equiv 1 \pmod{g}$. To encrypt a message M consisting of $n(n-1)/2$ elements of Z_m, fill in the upper triangular entries of a matrix U row by row with these elements, choose random diagonal entries relatively prime to m, and use the RSA equation $E(U) = U^e$ for encryption and $D(U) = U^d$ for decryption.

Example 7.1 We construct a trapdoor hash function

$$h : Z_{n+n(n-1)/2} \times Z_{n+n(n-1)/2} \longrightarrow Z_{n+n(n-1)/2}$$

for pairs (e_i, d_i) of such matrix exponents via

$$h(U_1, U_2) = U_1^{e_1} \cdot U_2^{e_2},$$

where the $n \times n$ upper triangular matrices U_1, U_2 are each formed from the $n + n(n-1)/2$ message entries by filling in the matrix row by row, diagonal entries included.

If all the diagonal entries of *both* such matrices are relatively prime to m, then for any upper triangular invertible matrix P

$$h((U_1^{e_1} \cdot P)^{d_1}, (P^{-1} \cdot U_2^{e_2})^{d_2}) = (U_1^{e_1} \cdot P) \cdot (P^{-1} \cdot U_2^{e_2}) = h(U_1, U_2)$$

and we have a solution to the Type II collision problem.

To make sure that the diagonal entries are relatively prime to m, one would like to apply an algorithm that re-assigns the elements of Z_m to Z_m^* without revealing the factorization of m. We do not know if this is possible.

[3] In [11], it is shown that for $m = \Pi\, p_i^{r_i}$, g is a divisor of $\mathrm{lcm}(\varphi(p_i^{r_i})p_i^{r_i})$.

The trapdoor feature to this hash is wholly dependent on the randomness of the diagonal entries since if the diagonal entries were fixed then, for example, $(U_1^{e_1} \cdot P)^{d_1}$, would almost surely not be an admissible message. It is therefore clear that this trapdoor hash is another very fragile one: Potential "improvements" such as commingling diagonal entries in order that some of the entries from U_1 are appropriated for the U_2 diagonal, and conversely, would destroy the trapdoor nature of this construction.

8 Conclusion

We have attempted to survey the spectrum of public-key cryptosystems for the purpose of constructing secure trapdoor hashing algorithms. We have exhibited our constructions, analyzed their strengths and weaknesses, and explored their ramifications. We have demonstrated that such constructions are theoretically possible but that they are likely to depend on the density of the encryption algorithm, and that they are most often fragile in a very explicit sense. Though not all the public-key cryptosystems we reviewed are considered in this paper — elliptic curve methods [3], methods grounded in class number fields, and finite group mapping methods [6] for example, all required computational overhead that made them totally inappropriate for adaptation to our hashing context — we believe those we have drawn from are a representative sample from which to extract ideas suitable to our task.

Whether we have indeed provided convincing evidence that it might be possible to conceal a public-key cryptosystem in a secure hash algorithm the reader will have to decide for himself or herself.

References

[1] R. Cooper and W. Patterson, A generalization of the knapsack algorithm using finite fields, *Cryptologia*, Volume VIII, Number 4, October 1984, 343–347.

[2] P. Guan, Cellular automaton public-key cryptosystem, *Complex Systems*, Vol 1, 1987, 51–57

[3] A. Menezes, *Elliptic Curve Public-Key Cryptosystems*, Kluwer, Norwell, MA, 1993.

[4] B. McKeever and S. Spence, SB-Dac: A study of the applications of finite field automorphisms to message digest algorithms, *preprint*.

[5] W. Patterson, *Mathematical Cryptology for Computer Scientists and Mathematicians*, Rowman & Littlefield, Totowa, NJ, 1987.

[6] M. Qu and S. Vanstone, New public-key cryptosystems based on factorizations of finite groups, *Advances in Cryptology — AUSCRYPT 92*, to appear.

[7] B. Schneier, *Applied Cryptography : protocols, algorithms, and source code in C*, Wiley, New York, 1994.

[8] J. Seberry and J. Pieprzyk, *Cryptography — An Introduction to Computer Security*, Prentice Hall, NJ, 1989.

[9] W. Stallings, *Protect your privacy : the PGP user's guide*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[10] W. Webb, A public key cryptosystem based on complementing sets, *Cryptologia*, Volume XVI, Number 2, April 1992, 177–181.

[11] V. Varadharajan and R. Odoni, Extension of RSA cryptosystems to matrix rings, *Cryptologia*, Volume IX, Number 9, April 1985, 140–153.