Remarks on MOBS and cryptosystems using semidirect products

Chris Monico

Department of Mathematics and Statistics
Texas Tech University
e-mail: c.monico@ttu.edu

Draft of September 1, 2021

Abstract

Recently, several cryptosystems have been proposed based semidirect products of various algebraic structures [5, 6, 4, 9]. Efficient attacks against several of them have already been given [7, 8, 11, 3, 2], along with a very general attack in [10]. The purpose of this note is to provide an observation that can be used as a point-of-attack for similar systems, and show how it can be used to efficiently cryptanalyze the MOBS system.

1 General semidirect product cryptosystems

In this section, we describe the general framework encompassing several recently proposed algebraic cryptosystems, including [5, 6, 4, 9], and give a general observation which applies to them all. That observation will be used in the next section to give a polynomial-time attack on the proposed MOBS system [9].

Suppose that G is a semigroup and S is a sub-semigroup of endomorphisms of G. One can define the semidirect product $G \rtimes S$ as the set $G \times S$ together with the operation

$$(g_1, \phi_1)(g_2, \phi_2) = (\phi_2(g_1)g_2,\ \phi_1 \circ \phi_2).$$

One can then build a Diffie-Hellman-like key exchange protocol as follows.

(i) Alice and Bob agree on an element $(g, \phi) \in G \rtimes S$.

(ii) Alice chooses a private integer a, computes $(g, \phi)^a = (A, \phi^a)$, and sends A to Bob.

(iii) Bob chooses a private integer b, computes $(g, \phi)^b = (B, \phi^b)$, and sends B to Alice.

(iv) Alice computes $K_A = \phi^a(B)A$.

(v) Bob computes $K_B = \phi^b(A)B$.

Since

$$(K_A, \phi^{a+b}) = (B, \phi^b)(A, \phi^a) = (g, \phi)^{b+a} = (A, \phi^a)(B, \phi^b) = (K_B, \phi^{a+b}),$$

it follows that $K_A = K_B$, so this is Alice and Bob's shared secret key, K. One also has that

$$\begin{aligned} A &= \phi^{a-1}(g)\phi^{a-2}(g)\cdots\phi(g)g, \\ B &= \phi^{b-1}(g)\phi^{b-2}(g)\cdots\phi(g)g, \quad \text{and} \\ K &= \phi^{a+b-1}(g)\phi^{a+b-2}(g)\cdots\phi(g)g. \end{aligned}$$

In general, it is not necessary for an attacker Eve to determine a or b to recover the shared key K. It would be sufficient for her to find an endomorphism ψ of G which commutes with ϕ and satisfies

$$\psi(g)A = \phi(A)g. \tag{1.1}$$

If she can find such an endomorphism, it follows that

$$\begin{aligned} \psi(B)A = \psi\left(\prod_{j=b-1}^{0} \phi^j(g)\right) A = \left(\prod_{j=b-1}^{0} \phi^j(\psi(g))\right) A &= \left(\prod_{j=b-1}^{1} \phi^j(\psi(g))\right) \psi(g)A \\ &= \left(\prod_{j=b-1}^{1} \phi^j(\psi(g))\right) \phi(A)g \\ &= \left(\prod_{j=b-1}^{2} \phi^j(\psi(g))\right) \phi(\psi(g)A)g \\ &= \left(\prod_{j=b-1}^{2} \phi^j(\psi(g))\right) \phi^2(A)\phi(g)g \\ &\;\;\vdots \\ &= \phi^b(A)B = K. \end{aligned}$$

2 MOBS

In [9], the authors propose the following. Let k be a positive integer and let $\mathcal{B}_k$ denote the semiring of bitstrings of length k (i.e., $\mathcal{B}_k = \mathbb{Z}_2^k$, as a set), together with the operations of bitwise OR and bitwise AND. It's easy to see that AND distributes over OR and both operations are associative, so $\mathcal{B}_k$ with these operations is indeed a semiring. Then G will be the multiplicative semigroup of $n \times n$ matrices over $\mathcal{B}_k$.

A permutation $\sigma \in S_k$ naturally acts on $\mathcal{B}_k$ by permuting the bits, and this extends to an action on G. The semigroup of endomorphisms S is taken as the symmetric group S_k; in fact, this is a group of automorphisms of G.

Suppose that g, ϕ, A, and B are as in the previous section with this choice of G and S. We will now show how to produce an endomorphism ψ which commutes with ϕ and satisfies (1.1). In fact, we will determine an integer α for which

$$\phi^\alpha(g)A = \phi(A)g.$$

First note that such an α necessarily exists, since Alice's integer a satisfies this.

Since ϕ is a permutation on $\{1, 2, \ldots, k\}$, we can determine its disjoint cycle decomposition $\phi = \sigma_1 \cdots \sigma_t$ with $\mathcal{O}(k)$ operations. Since the cycles $\sigma_1, \ldots, \sigma_t$ are disjoint, they commute, and so $\phi^\alpha(g)A = \phi(A)g$ if and only if

$$\left(\sigma_1^\alpha \cdots \sigma_t^\alpha\right)(g)A = \phi(A)g.$$

For each j, one can find an integer α_j for which $\sigma_j^{\alpha_j}(g)A$ agrees with $\phi(A)g$ in the bit positions corresponding to that cycle (i.e., the orbit of σ_j which has length greater than 1). This can be done with brute force by computing $gA, \sigma_j(g)A, \sigma_j^2(g)A, \ldots$ until such an α_j is found. This requires that we compute at most $|\sigma_j|$ permutation products and matrix products.

Then use the Chinese Remainder Theorem to find an integer α for which $\alpha \equiv \alpha_j \pmod{|\sigma_\mathrm{j}|}$ for all j. It follows that $\phi^\alpha(g)A = \phi(A)g$.

Since $|\sigma_1| + \cdots + |\sigma_t| \leq k$, we have to compute no more than k permutation products and matrix products. Since these operations are polynomial-time in the key size, and k is less than the key size, it follows that this is polynomial-time. The final Chinese Remainder Theorem calculation solves a system of congruences with moduli $|\sigma_1|, \ldots, |\sigma_t|$. If $N = \prod |\sigma_j|$, then the size of N is about $\log N = \sum \log |\sigma_j| \leq k$, and this can be done using $\mathcal{O}(\log^2 N) = \mathcal{O}(k^2)$ operations [1], so it is also polynomial-time.

We extended the Python code generously made available by the authors of [9] to implement this attack, and ran experiments for various values of k (of the same form they suggested, being a sum of the first several primes). For each indicated value of k, we used $n = 3$ (3×3 matrices) and generated 20 shared keys. We report the average wall-clock time to recover each shared key on a single core of an i7 processor at 3.10GHz.

k	Avg. time (seconds)
100	0.0878
197	0.2374
381	0.5325
791	1.7000

References

[1] Eric Bach and Jeffrey Shallit. *Algorithmic number theory, volume 1: efficient algorithms.* MIT Press, Cambridge, Massachusetts, 1996. URL: `http://www.math.uwaterloo.ca/ shallit/ant.html`.

[2] Christopher Battarbee, Delaram Kahrobaei, and Siamak F. Shahandashti. Cryptanalysis of semidirect product key exchange using matrices over non-commutative rings, 2021.

[3] Daniel Brown, Neal Koblitz, and Jason LeGrow. Cryptanalysis of 'make'. Cryptology ePrint Archive, Report 2021/465, 2021. `https://ia.cr/2021/465`.

[4] Dima Grigoriev and Vladimir Shpilrain. Tropical cryptography II: extensions by homomorphisms. *Comm. Algebra*, 47(10):4224–4229, 2019.

[5] Maggie Habeeb, Delaram Kahrobaei, Charalambos Koupparis, and Vladimir Shpilrain. Public key exchange using semidirect product of (semi)groups. In *Applied cryptography and network security. 11th international conference, ACNS 2013, Banff, AB, Canada, June 25–28, 2013. Proceedings*, pages 475–486. Berlin: Springer, 2013.

[6] Delaram Kahrobaei and Vladimir Shpilrain. Using semidirect product of (semi)groups in public key cryptography. In *Pursuit of the universal. 12th conference on computability in Europe, CiE 2016, Paris, France, June 27 – July 1, 2016. Proceedings*, pages 132–141. Cham: Springer, 2016.

[7] M. Kreuzer, A. D. Myasnikov, and A. Ushakov. A linear algebra attack to group-ring-based key exchange protocols. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security*, pages 37–43, Cham, 2014. Springer International Publishing.

[8] Chris Monico and Ayan Mahalanobis. A remark on MAKE - a matrix action key exchange. *CoRR*, abs/2012.00283, 2020.

[9] Nael Rahman and Vladimir Shpilrain. Mobs (matrices over bit strings) public key exchange. Cryptology ePrint Archive, Report 2021/560, 2021. `https://ia.cr/2021/560`.

[10] Vitaliĭ Roman'kov. A nonlinear decomposition attack:. *Groups Complexity Cryptology*, 8(2):197–207, 2016.

[11] Dylan Rudy and Chris Monico. Remarks on a tropical key exchange system:. *Journal of Mathematical Cryptology*, 15(1):280–283, 2021.