Cryptanalysis of Kowada-Machado key exchange protocol

M. Coutinho, T. C. de Souza Neto, R. de O. Albuquerque and R. T. de Sousa Júnior

***Abstract*—A non-interactive key exchange (NIKE) protocol allows N parties who know each other's public key to agree on a symmetric shared key without requiring any interaction. A classic example of such protocol for $N = 2$ is the Diffie-Hellman key exchange. Recently, some techniques were proposed to obtain a NIKE protocol for N parties, however, it is still considered an open problem since the security of these protocols must be confirmed. In a recent work, Kowada and Machado [1] proposed a protocol that solves the NIKE problem for N parties. However, this work found security problems in the proposed solution and implemented an efficient attack to their protocol demonstrating that their key-exchange scheme is insecure.**

***Keywords*—Key exchange, Cryptography, Cryptanalysis.**

I. INTRODUCTION

Key exchange schemes are very important in cryptography. A non-interactive key exchange (NIKE) protocol is designed to allow N parties to agree on a shared secret without requiring any interaction. Usually, N parties publish their public keys and then agree on a shared key k that is secret from any eavesdropper who only sees the public keys.

In 1976, Diffie and Hellman [2] revolutionized the field with a non-interactive key exchange protocol (NIKE) for 2 parties ($N = 2$). Since then, several other schemes were proposed for $N = 2$ using different techniques such as Elliptic Curves [3] and El-Gamal [4]. In 2004, Joux [5] was the first author who solved the problem for $N = 3$ using bilinear maps.

The development of NIKE protocols for an arbitrary number of parties N has been a research topic for several years in the cryptographic field and some of the greatest researches of the area have tried to solve this problem.

In 2013, Coron et. al. [6], introduced the first implementation of a key exchange for N parties using multi-linear maps and lattices, which requires an initial setup between all parties. In 2017, Boneh et. al. [7] used indistinguishability obfuscation to propose the first technique that seems to solve the NIKE problem for N parties.

In this context, Kowada and Machado [1] proposed a new protocol to solve the NIKE problem for N parties. Unfortunately, as this paper will show, their scheme is insecure as there is a very efficient attack to compute the shared secret based exclusively on the public keys.

This paper is organized as follows: in section II, presents the Kowada-Machado (KM) protocol. In section III, some number theory results used in the construction of the attack, are presented. In section IV the proposed attack is described, an actual implementation is presented and practical results are given. Finally, in section V the conclusions are presented.

M. Coutinho, Cybersecurity INCT Unit 6, Decision Technologies Laboratory - LATITUDE, Electrical Engineering Department (ENE), Technology College, University of Brasília (UnB), coutinho.stat@gmail.com

T. C. de Souza Neto, Cybersecurity INCT Unit 6, Decision Technologies Laboratory - LATITUDE, Electrical Engineering Department (ENE), Technology College, University of Brasília (UnB), tsouzaneto@gmail.com

R. de O. Albuquerque, Cybersecurity INCT Unit 6, Decision Technologies Laboratory - LATITUDE, Electrical Engineering Department (ENE), Technology College, University of Brasília (UnB), robson@redes.unb.br

R. T. de Sousa Júnior, Cybersecurity INCT Unit 6, Decision Technologies Laboratory - LATITUDE, Electrical Engineering Department (ENE), Technology College, University of Brasília (UnB), desousa@unb.br

II. KOWADA-MACHADO KEY EXCHANGE

Kowada and Machado [1] proposed a NIKE protocol involving N parties. The idea is similar to the Diffie-Hellman protocol [2]. Namely, it uses the exponentiation in a finite field and relies on the difficulty of the discrete logarithm problem. The difference is that the exponent is a quadratic function inspired on diophantine equations [8].

Basically, the public parameters $\alpha, \beta \in \mathbb{N}$ define the quadratic equation for all parties, $\delta \in \mathbb{N}$ is a dimension parameter and $y \in \mathbb{N}$ is a base. All these parameters are generated by one of the N parties or by a trusted third party. Considering $\varphi(x)$ as Euler's phi function and (x, y) as the greatest common divisor (GCD) of x and y, the Kowada-Machado protocol is described in Algorithm 1.

Algorithm 1 Key establishment

```
• Public parameters definition
    1) Choose $\delta \in \mathbb{N}$
    2) Choose $\beta \in \mathbb{N}$ such that $\varphi(\delta) \nmid \beta$
    3) Choose $\alpha \in \mathbb{N}$ such that $\varphi(\delta) \mid \alpha$
    4) Choose $y \in \mathbb{N}$ such that $y < \delta$ and $(y, \delta) = 1$
    5) Publish $\delta, y, \alpha, \beta$
• Key generation for each party
    1) Each party $i$ chooses a pair $(x_{a_i}, x_{b_i})$, such that
       $(x_{b_i}, \alpha) = 1$. This defines the private key.
    2) Each party calculates $\gamma_i = \alpha x_{a_i}^2 + \beta x_{b_i}$.
    3) Each party publishes its public key $\gamma_i$.
• Computing a shared secret
    1) Each party $i$ computes a shared secret using his
       private key $x_{b_i}$ and multiplying by the public keys
       of all other parties:
```

$$k = y^{\beta^{N-1} \prod_{t=1}^{N} x_{b_i}} \mod \delta$$

All parties can calculate the same secret using the fact that $y^{\alpha x} \equiv 1 \mod \delta$ since $\varphi(\delta) \mid \alpha$. In their paper [1], the authors also define two ways of generating the public parameters.

However, we will not address that here since it does not affect the proposed attack.

III. Modular equations

In this section, we present some number theory results that we will use as a basis of the attack. The interested reader can find more details on [9] or [10].

Consider the linear equation

$$ax \equiv b \mod m \tag{1}$$

Let $d = (a, m)$. It is known that if $d|b$, then this equation has d incongruent solutions modulo m. Additionally, given a solution to Eq. (1), it is possible to compute the remaining $d-1$ solutions from the first. Indeed, we have the following:

Lemma 3.1: Let x_0 be a solution to Eq. (1). Then

$$x_0 + \frac{m}{d}i$$

is also a solution of Eq. (1) for all $i = 0, ..., d-1$.

Proof: By definition, we have $ax_0 \equiv b \mod m$. Then, for each $i \in \{0, \ldots, d-1\}$, we have

$$a\left(x_0 + \frac{m}{d}i\right) \equiv ax_0 + a\frac{m}{d}i \mod m$$

Since $d|a$, it follows that $a\frac{m}{d}i$ is a multiple of m and therefore

$$a\left(x_0 + \frac{m}{d}i\right) \equiv ax_0 \equiv b \mod m$$

■

Under the hypothesis of Lemma (3.1), it is said that x_0 is a *fundamental solution* of Eq. (1). Next, we will show how to find a fundamental solution to Eq. (1). To do so, consider the following lemmas:

Lemma 3.2: Let x_0 be a solution of the equation

$$\frac{a}{d}x \equiv \frac{b}{d} \mod \frac{m}{d}$$

Thus, x_0 is a solution of Eq. (1).

Proof: By definition, there is a integer k such that

$$\frac{a}{d}x_0 = \frac{b}{d} + k\frac{m}{d}$$

Thus, $ax_0 = b + km$ and therefore x_0 is a solution of $ax \equiv b \mod m$. ■

Lemma 3.3:

$$x_0 = \left(\frac{a}{d}\right)^{-1}\frac{b}{d} \mod \frac{m}{d}$$

is a solution of the equation

$$\frac{a}{d}x \equiv \frac{b}{d} \mod \frac{m}{d}$$

Proof: Since $d = (m, a)$, it follows that $\left(\frac{m}{d}, \frac{a}{d}\right) = 1$. Consequently, $\frac{a}{d}$ has a inverse modulo $\frac{m}{d}$. A straightforward calculation concludes the proof. ■

We still need another important result that defines a way to work with exponents of modular equations.

Lemma 3.4: If $a \equiv b \mod \varphi(m)$, then

$$x^a \equiv x^b \mod m$$

Proof: We can write $a = k\varphi(m) + b$, for some integer k. Hence, from Euler's Theorem,

$$x^a = x^{k\varphi(m)+b} = x^{k\varphi(m)}x^b \equiv x^b \mod m.$$

■

IV. Attack

This section will define the proposed attack. Additionally, practical results are presented.

A. Theoretical framework

Consider the key establishment between N parties by Algorithm 1. In this case, define the private keys (x_{a_1}, x_{b_1}), (x_{a_2}, x_{b_2}), ..., (x_{a_N}, x_{b_N}) and the public keys $\gamma_1, \gamma_2, ..., \gamma_N$.

Note that the attacker has access to the public keys γ_i for $i = 1, ..., N$ and can compute

$$\sigma_i = \gamma_i \mod \varphi(\delta) \equiv \beta x_{b_i} \mod \varphi(\delta) \tag{2}$$

From now on, we will consider that $d = (\beta, \varphi(\delta))$.

Lemma 4.1: The solution to the system

$$\begin{aligned} \sigma_1 &\equiv \beta x_{b_1} \mod \varphi(\delta) \\ \sigma_2 &\equiv \beta x_{b_2} \mod \varphi(\delta) \\ &\vdots \\ \sigma_N &\equiv \beta x_{b_N} \mod \varphi(\delta) \end{aligned} \tag{3}$$

has the form

$$\left(x_{0_1} + \frac{\varphi(\delta)}{d}j_1, x_{0_2} + \frac{\varphi(\delta)}{d}j_2, \ldots, x_{0_N} + \frac{\varphi(\delta)}{d}j_N\right),$$

where $j_i = 0, 1, \ldots, d-1$ for all $i \in \{1, \ldots, N\}$ and x_{0_i} is a fundamental solution of $\sigma_i \equiv \beta x_{b_i} \mod \varphi(\delta)$.

Proof: It follows directly from Lemma 3.1. ■

Lemma 4.2: The exponent

$$\beta^{N-1}\prod_{i=1}^{N} x_{b_i} \mod \varphi(\delta)$$

is invariant under the choice of the fundamental solution.

Proof: Let $x_{0_i} + \frac{\varphi(\delta)}{d}j_i$ be any solution of $\sigma_i \equiv \beta x_{b_i} \mod \varphi(\delta)$. Then

$$\beta^{N-1}\prod_{i=1}^{N} x_{b_i} = \beta^{N-1}\prod_{i=1}^{N}\left(x_{0_i} + \frac{\varphi(\delta)}{d}j_i\right)$$

That is,

$$\begin{aligned} \beta^{N-1}\prod_{i=1}^{N} x_{b_i} = {} & \beta^{N-1}x_{0_1}x_{0_2}\ldots x_{0_N} \\ & +\beta^{N-1}\frac{\varphi(\delta)}{d}(j_1x_{0_2}\ldots x_{0_N} \\ & +x_{0_1}j_2\ldots x_{0_N} + \cdots + x_{0_1}\ldots x_{0_{N-1}}j_N) \\ & +\ldots \\ & +\beta^{N-1}\left(\frac{\varphi(\delta)}{d}\right)^{N-1}(x_{0_1}j_2\ldots j_N \\ & +j_1x_{0_2}\ldots j_N + \cdots + j_1\ldots j_{N-1}x_{0_N}) \\ & +\beta^{N-1}\left(\frac{\varphi(\delta)}{d}\right)^{N} j_1j_2\ldots j_N \end{aligned} \tag{4}$$

Note that since $d \mid \beta$ and $d \mid \varphi(\delta)$ then all terms, except the first, are multiples of $\varphi(\delta)$. Therefore, it follows that

$$\beta^{N-1}\prod_{i=1}^{N} x_{b_i} \equiv \beta^{N-1}x_{0_1}x_{0_2}\ldots x_{0_N} \mod \varphi(\delta) \tag{5}$$

■

This result is the cryptanalysis of KM key exchange protocol since the attacker can calculate the shared secret from the public parameters and keys in a very efficient way. In the next section we will define specifically the attack as an algorithm and will show an actual implementation and its performance.

B. Algorithm

Algorithm 2 details the proposed algorithm for the cryptanalysis of the KM protocol. The attack is based on the results of Lemmas 4.1 and 4.2. The algorithm is extremely efficient since it only uses basic modular operations, the Euclidean algorithm to compute d, and the Extended Euclidean algorithm to compute I.

Algorithm 2 Cryptanalysis of KM protocol

- **Definitions**
  1) The attacker has access to δ, α, β, since they are all public parameters
  2) The attacker has access to the public key of each one of the N parties $\gamma_1, \dots, \gamma_N$
- **Finding the fundamental solutions**
  1) Compute $\sigma_i = \gamma_i \mod \varphi(\delta)$ for $i = 1, ..., N$ obtaining the system
  $$\begin{aligned}\sigma_1 &\equiv \beta x_{b_1} \mod \varphi(\delta)\\ \sigma_2 &\equiv \beta x_{b_2} \mod \varphi(\delta)\\ &\vdots\\ \sigma_N &\equiv \beta x_{b_N} \mod \varphi(\delta)\end{aligned}$$
  2) Compute $d = (\beta, \varphi(\delta))$
  3) Compute $I = \left(\frac{\beta}{d}\right)^{-1} \mod \frac{\varphi(\delta)}{d}$
  4) Compute the fundamental solutions from
  $$\begin{aligned}x_{0_1} &= I\frac{\sigma_1}{d} \mod \frac{\varphi(\delta)}{d}\\ x_{0_2} &= I\frac{\sigma_2}{d} \mod \frac{\varphi(\delta)}{d}\\ &\vdots\\ x_{0_N} &= I\frac{\sigma_N}{d} \mod \frac{\varphi(\delta)}{d}\end{aligned}$$
- **Compute the shared secret**
  1) The attacker computes
  $$S = \beta^{N-1} x_{0_1} x_{0_2} \dots x_{0_N} \mod \varphi(\delta)$$
  2) Calculate the shared secret
  $$k = y^S \mod \delta$$

The attack was implemented using the RELIC tool-kit [11], a cryptographic library with emphasis on efficiency and flexibility. The source code is given in Appendix A.

C. Practical example

To illustrate the attack we will use the example provided in the original work [1]. Let $\delta = 33$, $\alpha = 20$, $\beta = 455$, $y = 10$ and $\varphi(\delta) = 20$. Suppose there are two parties with public keys $\gamma_1 = 20755$ and $\gamma_2 = 12885$. In their work, the authors compute the shared secret as $k = 10$. Now, we will show how the attacker can compute this shared secret using Algorithm 2.

1) The attacker computes
$$\begin{aligned}\sigma_1 &= 15 \equiv 20755 \mod 20\\ \sigma_2 &= 5 \equiv 12885 \mod 20\end{aligned}$$
2) The attacker computes $d = (455, 20) = 5$
3) The attacker computes $I = 3 \equiv 91^{-1} \mod 4$
4) The attacker computes the fundamental solutions
$$\begin{aligned}x_{0_1} &= 1 \equiv 3 * \tfrac{15}{5} \mod 4\\ x_{0_2} &= 3 \equiv 3 * \tfrac{5}{5} \mod 4\end{aligned}$$
5) Calculate $S = 5 \equiv 455 * 3 * 1 \mod 20$
6) Finally, the attacker obtain $k = 10 \equiv 10^5 \mod 33$

As expected, the attacker found the shared secret key $k = 10$.

		Key size in bits		
		1024	2048	4096
N	2	0.0044	0.0215	0.1313
	3	0.0054	0.0215	0.1469
	5	0.0053	0.0220	0.1360
	10	0.0053	0.0231	0.1569

TABLE I
TIME IN SECONDS TO EXECUTE THE ATTACK FOR KEY SIZES OF 1024, 2048 AND 4096 BITS FOR DIFFERENT NUMBER OF PARTIES.

D. Computational complexity

The proposed attack was tested in a single machine against big keys. It was verified that the attack is extremely fast, the results are presented in Table I. The key sizes of 1024, 2048 and 4096 were used because these are common values for the DH protocol being infeasible to calculate the discrete log problem as an attack as discussed in [1].

Note that the attack is extremely fast, demanding less than 1 second even for keys of 4096 bits. In fact, the algorithm only uses the Euclidean algorithm, which is known to have polynomial complexity [12]. In Figure 1 is possible to note that the proposed algorithm has polynomial complexity in the number of bits of the key.

V. Conclusion

In this work we presented the cryptanalysis of the Kowada-Machado key exchange protocol. Although the KM scheme solves the non-interactive key exchange problem for N parties, it does so in an insecure way.

Using the proposed attack and its implementation it is possible to recover the shared secret based only on the public keys in a very efficient way. Effectively, the attack can recover the shared secret in a few seconds even for keys of a very large size.

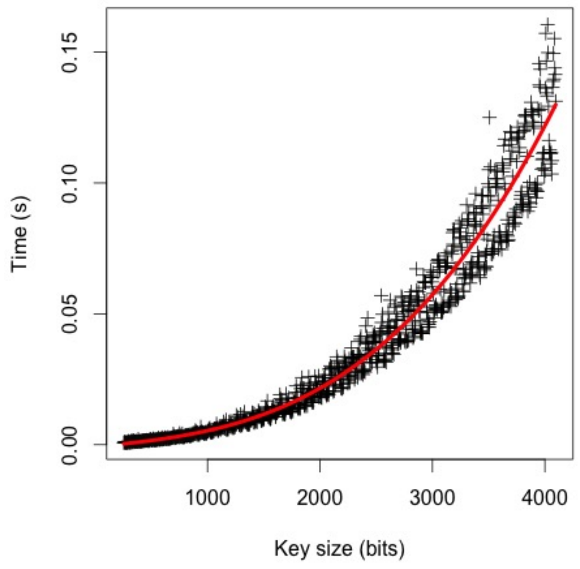

Figure 1. The complexity of the attack. In black, the execution times obtained for different key sizes. In red, polynomial $y = 1.610^{-12}x^3 + 3.610^{-10}x^2 + 4.210^{-6}x - 7.9 * 10^{-4}$, showing that the complexity is polynomial in the number of bits.

References

[1] L. A. B. Kowada and R. C. S. Machado, "Esquema de acordo de chaves de conferência baseado em um problema de funções quadráticas de duas variáveis," *XVII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg*, 2017.

[2] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[3] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 119–134, 2003.

[4] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[5] A. Joux, "A one round protocol for tripartite diffie–hellman," in *International algorithmic number theory symposium*, pp. 385–393, Springer, 2000.

[6] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Practical multilinear maps over the integers," in *Advances in Cryptology–CRYPTO 2013*, pp. 476–493, Springer, 2013.

[7] D. Boneh and M. Zhandry, "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation," *Algorithmica*, vol. 79, no. 4, pp. 1233–1285, 2017.

[8] L. J. Mordell, *Diophantine equations*, vol. 30. Academic Press, 1969.

[9] S. Shokranian, M. Soares, and H. Godinho, *Number Theory*. UnB, 1994.

[10] K. Ireland and M. Rosen, *A classical introduction to modern number theory*, vol. 84. Springer Science & Business Media, 2013.

[11] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography." https://github.com/relic-toolkit/relic.

[12] D. E. Knuth, *The art of computer programming: sorting and searching*, vol. 3. Pearson Education, 1998.

Appendix

```
void cryptanalysisKowadaMachado(bn_t k,
   bn_t *Pub, bn_t beta, bn_t delta,
   bn_t phi, bn_t y, int N )
{
    bn_t x0, d, I, phiOverD, betaOverD;
    bn_t sigma, S;

    bn_null(x0);
    bn_new(x0);
    ...
    bn_null(S);
    bn_new(S);

    bn_gcd_basic(d, beta, phi);
    bn_div(phiOverD, phi, d);
    bn_div(betaOverD, beta, d);

    //I = betaOverD^{-1} mod phiOverD
    bn_gcd_ext_basic(betaOverD, I, NULL,
    betaOverD, phiOverD);
    if (bn_sign(I) == BN_NEG) {
        bn_add(I, I, phiOverD);
    }

    //S = beta^{N-1}
    bn_mxp_dig(S, beta, N-1, phi);

    //For each Public Key, find
    //the solution x0 and multiply by S.
    for(int i = 0; i < N; i++)
    {
        bn_mod_basic(sigma, Pub[i], phi);
        bn_div(sigma,sigma,d);
        bn_mul_basic(x0, I, sigma);
        bn_mod_basic(x0, x0, phiOverD);

        bn_mul_basic(S, S, x0);
        bn_mod_basic(S, S, phi);
    }

    //k = y^{S} mod delta
    bn_mxp_basic(k, y, S, delta);

    bn_free(x0);
    ...
    bn_free(S);
}
```