5-2011

A Survey of Modern Mathematical Cryptology

Kenneth Jacobs
The University of Tennessee, ken.s.jacobs@gmail.com

Survey of Modern Mathematical Cryptology

Kenneth Jacobs

April 15, 2011

Abstract

Much of modern applied mathematics, and in particular modern applied algebra, has focused on creating algorithms by which two parties can securely communicate information. Numerous such algorithms have been published in the past thirty years; some have been shown as insecure whereas others have withstood scrutiny for many years. A class of these algorithms, known as key agreement protocols, are of particular importance because of their utility in secure, rapid encryption. In this paper, we will explore some of the ideas behind some public-key cryptosystems; in particular, we will analyze an algorithm that was proposed in [2].

1 Background

Cryptology is the study of ways in which two parties can securely communicate information. Over the past century, the advent and rapid growth of digital communication has led to mathematical algorithms for ensuring the security of encrypted messages. The practice of encoding messages, however, dates back nearly as far as does written communication; Egyptians, Romans, and feudal lords have used methods to keep their correspondence secret. The most recent developments in cryptology mark a significant departure from the traditional methods of encoding messages, and seem to offer higher levels of security.

In the cryptosystems considered here, an *encryption algorithm* is a map $e : A \times K \to A$, where A is a set called an *alphabet*, and K is a set called the *keyspace*. A *decryption algorithm* (relative to e) is a map $d : A \times K \to A$, where the alphabet and keyspace are the same as for e. Further, for each key $k_1 \in K$, there must exist $k_2 \in K$ satisfying $d(e(x, k_1), k_2) = x$. The collection (A, K, e, d) forms a *cryptosystem*.

A very simple example of this is the affine cryptosystem; let $A = F^d$, K be the set $GL_d(F) \times F^d$, and define a map $e_a : F^d \times K \to F^d$ by $e_a(x, (K, k)) = Kx + k$. The decryption algorithm relative to e_a is given $d_a(x, (K^{-1}, K^{-1}k))$; it satisfies $d_a(e_a(x, (K, k)), (K^{-1}, K^{-1}k)) = x$. Note that in the affine system, a person wishing to decrypt a message must know the same information as the person who encrypted the message. For this reason, the affine cipher is called a symmetric cryptosystem.

Definition 1 (Symmetric Cryptosystem)**.** A cryptosystem in which they key k_1 used for encryption and the key k_2 used for decryption are known to both communicating parties.

Examples of symmetric cryptosystems are the Caesar Shift cipher from the Roman empire, the Viginère cipher, and the ciphers produced by the Enigma machines from the Second World War. When implementing these or other symmetric cryptosystems, the two communicating parties need a way by which they can establish a common key, and this process becomes one of the most important aspects of secure encryption. Simply using the same key for each encrypted message allows an eavesdropper to employ crypto-analytic techniques such as frequency analysis to determine the key and decrypt the secure communication. Even recycling a few keys on some sort of pattern does not afford a high level of security.

What is necessary is for two parties to have a secure way to create a key that they can use in their encryption and decryption. Now, if the two parties must meet and agree upon a key for each time they wish to communicate, then they might as well share whatever secret message they have to share in such meetings; therefore, the typical assumption is that the symmetric keys must be created 'on-the-fly'. Of course, they cannot use symmetric ciphers with pre-determined keys to create such keys, since there is no particular advantage to using one cipher to create a key for another cipher other than simply layering the overall cryptosystem; even layered systems are subject to similar crypto-analytic techniques mentioned above.

In the ideal situation, then, the two communicating parties would be able to create a key without using some other symmetric protocol. This leads to the notion of an asymmetric key agreement protocol; in these protocols, each of the communicating parties have some piece of private information and share some public information.

Definition 2 (Asymmetric Key Agreement Protocol)**.** Let A and B be parties wishing to establish a symmetric key; let K_1 be a key known only to A and K_2 a key known only to B. Let K be all other public information shared between these two parties. Then an algorithm by which A and B can compute a common value K^* using the public information K and their respective private information is called an aysmmetric key agreement protocol.

Using their private data and the available public data, each party is able to agree upon a key that can then be used in a symmetric cryptosystem. Perhaps an example will be illustrative.

2 Example of Asymmetric Protocol

Let's consider the group $G = \mathbb{Z}_{19} - \{0\}$, the invertible elements of the finite field with 19 elements. All calculations and values will be in this field.

Suppose that two parties, Alice and Bob, wish to develop a key to use in a symmetric encryption protocol, but are unable to meet in person to agree upon a key. Alice initiates the protocol by declaring a random element, say $q = 5 \in \mathbb{Z}_{19}$, to the public, perhaps publishing it on an internet blog.

Alice's Private Information:	Public Information:	Bob's Private Information:
	field = $\mathbb{Z}_1 9$ q=5	

When Bob and Alice decide to communicate, they each choose an integer to serve as their private information; we'll suppose that Alice chooses $a = 17$ while Bob chooses $b = 30$.

Alice's Private Information:	Public Information:	Bob's Private Information:
a=17	field = $\mathbb{Z}_1 9$	b=30
	q=5	

Alice now uses the public information and her private information to compute her public key, $K_a = 5^{17}(mod\ 19) = 4(mod\ 19)$; this is published to the public. Likewise, Bob computes and publishes $K_b = 5^{30}(mod\ 19) = 11$.

Alice's Private Information:	Public Information:	Bob's Private Information:
a=17	field = $\mathbb{Z}_1 9$	b=31
	q=5	
	$K_a = 4$	
	$K_b = 11$	

Alice now computes $(K_b)^a(mod\ 19) = 11^{17}(mod\ 19) = 7(mod\ 19)$ and Bob computes $(K_a)^b(mod\ 19) = 4^{30}(mod\ 19) = 7(mod\ 19)$; note that Alice and Bob have arrived at the same element of $\mathbb{Z}_{19}$, which becomes their shared key that they can then use in a symmetric protocol.

One immediate question arises: why it is necessary to utilize such a protocol only for creating a key and not for communicating a message? An answer to this question is that it is often much easier to implement symmetric protocols rather than asymmetric protocols; therefore, the former are preferred for communicating long messages whereas the latter are preferred for quickly creating keys to use in symmetric protocols. Since the key that two parties are using in each symmetric encryption changes with each message, traditional techniques such as frequency analysis are no longer reliable methods for decoding encrypted communications.

3 Modern Protocols and Algorithms

Note that, in the above protocol, in order to compute the shared key K, it would be sufficient for an eavesdropper to know either Alice's secret key a or Bob's secret key b; thus, the security of this protocol depends on the difficulty finding these numbers based on the public information. In particular, solving the equation $q^a = K_a$ for the exponent a must be 'hard'; the task of computing this exponent is called the discrete logarithm problem:

Definition 3 (Discrete Logarithm Problem)**.** Let G be a group and let $\alpha \in G$. Suppose that β is in $< \alpha >$, the cyclic subgroup generated by α. Then there exists an exponent l, unique up to the order of α, satisfying $\beta = \alpha^l$. If one is given α and β, the task of computing l is called the Discrete Logarithm Problem (DLP).

The discrete logarithm problem in a *general group* G is algorithmically 'hard', which is to say that the best known algorithms for computing the exponent run in time $O(2^{\frac{N}{2}})$, where N is the number of bits necessary to encode an element of G on a computer [5]; typically,

$N = \log_2(|G|)$. There are, however, certain groups whose properties yield algorithms that run faster than $O(2^{\frac{N}{2}})$. For example, the index calculus algorithm can solve the DLP in the multiplicative group of certain finite fields in subexponential time[5].

Another choice of platform groups would be the general linear $GL(F)$ group over some finite field F; however, [4] provides a polynomial-time reduction of this protocol to the discrete logarithm over several finite fields, making this approach less efficient than discrete logarithms over finite fields. Perhaps the most popular choice of groups for DLP-based protocols is $G = E$, where E is an elliptic curve. We will not discuss such protocols nor algorithms for their solution here.

There also exist key agreement protocols that are not based on the DLP. One such protocol relies on the difficulty of the *conjugacy search problem*, which is the task of computing a group element x satisfying xgx^{-1}, assuming that one is given $y = xgx^{-1}$ and g. In some groups, such as the general linear group, this protocol offers minimal security, since linear algebraic techniques easily give solutions to these types of equations. In other platform groups, however, it is difficult to solve this problem. There is also the *word equivalency problem*, which is to decide when two words $a_1a_2a_3...a_n$ and $b_1b_2b_3...b_k$ in a group presentation are equivalent. Both the conjugacy search problem and the word equivalency problem have been employed and analyzed in a number of situations; [6] contains a good overview of these implementations and analyses.

4 Sakalauskas, Tvarijonas and Raulynaitis Key Agreement Protocol (STRKAP)

We now turn to a modification of the DLP over matrix groups that was proposed by Sakalauskas, Tvarijonas and Raulynaitis in 2007 [2]. While the DLP alone over the matrix group offers little additional security, the proposed protocol combines the DLP with the conjugacy search problem in the general linear group. It goes as follows:

1. Alice chooses an infinite, non-commutative group Γ, a field F, a representation $\rho : \Gamma \to GL_d(F)$. In Γ, Alice also chooses two commuting subgroups G_1 and G_2, which are mapped $\mathcal{A} = \rho(G_1)$ and $\mathcal{B} = \rho(G_2)$. Note that it is not necessary for F to be a finite field. All of this information is public.

2. Alice chooses an element $q \in \Gamma$ and a vector $a \in F^d$ and publishes these as public information.

3. To create her public key, Alice chooses a private element $\alpha \in G_1$ and an integer j. She computes $\rho(\alpha \cdot q \cdot \alpha^{-1}) = AQA^{-1}$ and then raises this to the j^{th} power. This matrix $K_a = AQ^jA^{-1}$ becomes her public key.

4. Bob now creates his public key by choosing $\beta \in G_2$ and an integer s; his key will be $K_b = BQ^sB^{-1}$, where $B = \rho(\beta)$.

5. Alice computes the matrix $K = (AK_bA^{-1})^j = A(BQ^sB^{-1})^jA^{-1} = (AB)Q^{sj}(AB)^{-1}$; Bob can compute the same matrix via K_a.

6. Each party can now compute the session key, which is $Ka \in F^d$.

The protocol is summarized in Table 1:

Alice's Private Data	Public Data	Bob's Private Data
A	Q, ρ, $\mathbb{F}_{p^k}$	B
j	$\mathcal{A}$, $\mathcal{B}$	s
	AQ^jA^{-1}	
	BQ^sB^{-1}	

Table 1: Key = $((AB)Q^{s+j}(AB)^{-1})a$

According to this protocol, the number of bits in the key, N, is given by $N = d\log_2(p^k)$, since the key is a vector in $(\mathbb{F}_{p^k})^d$. In the original publication of the key agreement protocol, several additional recommendations were made. First, the authors suggest using $\mathbb{F}_{2^k}$ as the field because it is easy to implement on a computer. Additionally, the authors suggest using an irreducible representation ρ, or at least a representation in which the matrix $\rho(q) = Q$ is irreducible; the assumption is that if Q has invariant subspaces then the protocol can be reduced to the DLP over multiplicative groups of finite fields. We shall address this final assumption in the next section.

5 Analysis

To begin our analysis of the STRKAP, we look at matrix properties which are invariant under conjugation, and the first (and most promising) is the collection of eigenvalues. We know that, for a general matrix Q having eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_k\}$, the eigenvalues of AQA^{-1} are $\{\lambda_1, \lambda_2, ..., \lambda_k\}$ when A is a non-singular matrix, and that the eigenvalues of Q^j are $\{\lambda_1^j, \lambda_2^j, ..., \lambda_k^j\}$. Since the eigenvalues of Q are the roots of its characteristic polynomial $c(x) \in \mathbb{F}_{p^k}[x]$, we have that each of the eigenvalues are elements of a finite field; in the case of the STRKAP, we know that these roots are in $\mathbb{F}_{p^{kd}}$, since Q is irreducible. Ideally, then, we can imitate the algorithm given in [4] to reduce to a DLP over the finite field $\mathbb{F}_{p^{kd}}$. Yet an example (given in the appendix) shows that an attempt to apply this algorithm to the STRKAP not only fails to work, but it may actually return an incorrect value that could easily be mistaken for the correct solution!

The reason that such algorithms do not work is that the conjugation by A induces a permutation of the eigenvalues, and so when we attempt to match eigenvalues of Q to eigenvalues of AQ^jA^{-1}, we are not guaranteed that the 'first' eigenvalue of Q corresponds to the 'first' eigenvalue of AQ^jA^{-1}. The algorithm used in [4] was never intended to address such permutations, since it relies on the fact that the same matrix can put both Q and Q^j into a block-diagonal form with the eigenvalues on the diagonal.

Because of the permutations induced by A, there is also no particular need to have Q be irreducible, since lower-dimension invariant subspaces will be permuted, and if d is sufficiently large and the dimension of each of these subspaces are relatively similar (ideally, all will have exactly the same dimension), it would be just as difficult to sort out the permutations as in the irreducible case.

6 Trial Exponentiation

We may begin by naïvely computing sequential powers of Q in an effort to find j, and in fact this approach reveals why the key must be the vector $Ka \in (\mathbb{F}_{p^k})^d$ rather than the matrix K. First, we present the algorithm:

1. For $i = 1, 2, ...|Q|$

   [a.] Compute the i^{th} power of $Q = Q \cdot Q^{i-1}$

   [b.] Solve the matrix equation $XQ^iX^{-1} = AQ^jA^{-1}$ and $YQ^iY^{-1} = BQ^sB^{-1}$ for solution pairs (i_X, X) and (i_Y, Y)

   [c.] Check if any solutions X commute with Bob's subgroup; check if any solutions Y commute with Alice's subgroup.

   [d.] Once a valid X or a valid Y has been found, continue the algorithm only for the other unknown matrix, and iterate until both solution pairs have been found.

There are several things that must be noted about the above algorithm. First, note that once we obtain pairs (i_X, X) and (i_Y, Y) such that $XQ^{i_X}X^{-1} = AQ^jA^{-1}$ and $YQ^{i_Y}Y^{-1} = BQ^sB^{-1}$, we can form the key as $K = BAQ^{js}(BA)^{-1} = B(AQ^jA^{-1})^sB^{-1} = B(XQ^{i_X}X^{-1})^sB^{-1} = X(BQ^sB^{-1})^{i_X}X^{-1} = X(YQ^{i_Y}Y^{-1})^{i_X}X^{-1} = XYQ^{i_Xi_Y}Y^{-1}X^{-1}$.

There also arises the question of how to perform the commutivity checks in step 1c; it turns out that it is unnecessary to perform the commutivity checks on the whole of $\mathcal{A}$ or $\mathcal{B}$ (which would result in a complexity polynomial in the order of these two groups); rather, we can perform the commutivity checks on the generators of these groups. For the remainder of this paper, we will let γ be the sum of the number of generators of $\mathcal{A}$ and the number of generators of $\mathcal{B}$. Step 1c, then, takes $O(\gamma)$ steps per solution X or Y. Since the solution space to a linear equation of the form $JX = XM$ is of dimension d [3], we will have at most $O(d\gamma)$ commutivity checks.

The only remaining question is how big $|Q|$ can possibly be. For this, we utilize the fact that Q is irreducible to realize that it has all of its eigenvalues in $\mathbb{F}_{p^{kd}}$. With this in mind, we can obtain two order bounds of Q.

First, let J be a Jordan block for Q, whereby $J = \lambda I + S$, where $\lambda \in \mathbb{F}_{p^{kd}}$, I is the $d \times d$ identity matrix, and S is the matrix with 1's on the first superdiagonal and zero's elsewhere. Then by the binomial theorem, we have that $J^k = (\lambda I + S)^k = \sum_{r=0}^{k} \binom{k}{r} \lambda^r I S^{k-r}$. Now, whenever we raise J to a power of p, we have that $\binom{p^n}{r} = 0$ for all $r \neq 0, p^n$, since the field $\mathbb{F}_{p^{kd}}$ has characteristic p. Therefore, $J^p = \lambda^p I + \lambda^0 S^p$.

Letting p_d be the first power of p that is greater than d, we have that S^{p_d} is the zero matrix, and hence J^{p_d} is the diagonal matrix with λ^{p_d} on the diagonals. Since $\lambda \in \mathbb{F}_{p^{kd}} - \{0\}$, we have that $(\lambda^{p_d})^{p^kd-1} = 1$. Finally, we will have that $(J^{p_d})^{p^{kd}-1}$ is the identity matrix. Since this happens regardless of which Jordan block we choose for Q, we are guaranteed that $(Q^{p_d})^{p^{kd}-1}$ is the identity matrix. Therefore, if we have an algorithm that checks every power of Q, it will have at most $(p_d)(p^{kd} - 1)$ iterations.

Now if $p_d = p$, then we will iterate the outermost look at most $p^{kd+1} - p$ times, giving an overall complexity estimate of $O(p^{kd+1}(f(N) + d^2\gamma))$, where $f(N)$ is a polynomial function

in the number of bits N that accounts for the miscellaneous linear algebra step 1b. Since $N = d\log_2(p^k)$, we have that this complexity is $O(\gamma 2^{N(\frac{1}{kd}+1)})$.

If, however, $p_d \geq p^2$, then we know that $p < d$ and that $\frac{1}{p}p_d \leq d$, so that $p_d \leq d^2$. In this case, then, we iterate the outermost loop at most $d^2(p^{kd}-1)$ times, giving an overall complexity estimate of $O(d^2\gamma 2^N) = O(\gamma 2^N)$.

After all of this work, it appears that we have only found an exponential, and therefore useless, algorithm. But there are many insights to gain from this algorithm. First, the presence of γ in the complexity estimate is very unsettling, for there is no way to know how large this could be. Even if γ is subexponential, this naïve method is highly inefficient, since generic algorithms can obtain complexity estimates as low as $O(2^{\frac{N}{2}})$, which is lower than our algorithm in the case that $p > d$. However, if:

- the STRKAP protocol were keyed on the matrix K rather than the vector Ka
- γ is subexponential in N
- $p_d \geq p^2$, whereby $p_d \leq d^2$,

then the cryptanalysis would give a different result: with K as the key, $N = d^2\log_2(p^k)$, whereby we obtain a complexity estimate of $O(\gamma 2^{\frac{N}{d}})$, and thus for $d > 2$ we would have an algorithm that runs faster than $O(2^{\frac{N}{2}})$, which is the time for generic algorithms [5].

In all, this naïve approach remains unsettling. To determine γ we must know something on the order of commuting subgroups of $GL(F)$; new research in commuting matrices that may help provide further information can be found in [1]. What this approach does illustrate, however, is how changing the key that a protocol generates may make a significant difference in the complexity estimate of a proposed cryptanalysis, just as did the difference between taking Ka and K as the keys in this protocol.

7 Another Partial Algorithm

Since the matrices A and B are the only things preventing us from using the algorithm in [4] to decode the STRKAP, it seems reasonable to attempt to remove these first. In the case that Q has a complete set of eigenvectors, we might be able to solve for the matrix A as follows:

Since Q has a complete set of eigenvalues, it must also have a complete set of nonzero eigenvectors $\{v_1, v_2, ..., v_d\}$ that form a basis for the space $(\mathbb{F}_{p^{kd}})^d$. Likewise, we know that AQ^jA^{-1} has a complete set of nonzero eigenvectors $\{w_1, w_2, ..., w_d\}$. We know further that the eigenvectors for Q^j are precisely those of Q, for Q has a complete set of eigenvectors and $Q^jv_i = \lambda_i^j v_i$ for all $i \in [d]$.

To be an eigenvector of AQ^jA^{-1} is to say that $AQ^jA^{-1}w_i = \gamma w_i$ (where γ is the j^{th} power of some eigenvalue of Q). This is to say that $Q^j(A^{-1}w_i) = \gamma(A^{-1}w_i)$, hence $A^{-1}w_i$ is an eigenvalue of Q^j. But as was mentioned above, we have that the eigenvectors of Q^j are eigenvectors of Q, hence $A^{-1}w_i = c_{\sigma(i)}v_{\sigma(i)}$, where σ is a permutation of $[d]$ and $c_{\sigma(i)}$ is in $\mathbb{F}_{p^{kd}}$. Equivalently, we have that $c_{\sigma(i)}Av_{\sigma(i)} = w_i$. The constant $c_{\sigma(i)}$ is present because A may scale vectors; the permutation σ is present because A may also permute these vectors. We now define several matrices:

$$V = \begin{pmatrix} | & | & & | \\ v_{\sigma(1)} & v_{\sigma(2)} & \dots & v_{\sigma(d)} \\ | & | & & | \end{pmatrix}$$

$$C = \begin{pmatrix} c_1 & 0 & 0 & \dots & 0 \\ 0 & c_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & c_d \end{pmatrix}$$

and

$$W = \begin{pmatrix} | & | & & | \\ w_1 & w_2 & \dots & w_d \\ | & | & & | \end{pmatrix}$$

Note that the matrices V and W are invertible, for their columns form a basis of the vector space. If we let P be the permutation matrix corresponding to $(\sigma)^{-1}$, then we have that $A^{-1}W = VCP$, i.e. $A = V^{-1}C^{-1}P^{-1}W = A$; letting X be the matrix $C^{-1}P^{-1}$, we have the matrix equation $A = V^{-1}XW$.

Now this matrix equation has two unknown matrices, A and X, and so in general is not helpful in computing A. However, we know that in the STRKAP, A commutes with elements of Bob's subgroup. By choosing any matrix B from this subgroup, we obtain that $AB = BA$, i.e. $(V^{-1}XW)B = B(V^{-1}XW)$. Letting $J = VBV^{-1}$ and $M = WB^{-1}W^{-1}$, we obtain the equation $JX = XM$; since we know J and M, we can solve this matrix equation using linear algebra techniques to come up with a class of matrices satisfying this equation. We know that at least one such solution, namely $C^{-1}P^{-1}$ must exist; however, there is no guarantee that this solution is unique; in fact, there will necessarily be more than one solution to this equation since J and M have the same eigenvalues [3]. Therefore, the only time that this algorithm will work is when there is only one solution (up to a constant multiple in $\mathbb{F}_{p^{kd}}$) of the form $C^{-1}P^{-1}$.

If such a solution X exists, we can compute $A = V^{-1}XW$ and reduce to the DLP in the cyclic group generated by Q without worrying about the possibility of eigenspaces being permuted. Moreover, since all of these eigenvalues are found in $\mathbb{F}_{p^{kd}}$, the reduction is to a single discrete logarithm problem in this finite field.

We summarize the algorithm as follows:

1. Obtain the eigenvalues of Q and AQ^jA^{-1}:
   - [a.] Compute the characteristic polynomials of Q and AQ^jA^{-1}
   - [b.] Factor these over the field $\mathbb{F}_{p^{kd}}$
2. Compute the corresponding eigenvectors of Q and AQ^jA^{-1}
3. Arrange these eigenvectors into arrays V and W and compute $A = WXV^{-1}$ by solving the equation $JX = XM$
4. Compute the discrete logarithm of Q^j base Q

This algorithm has the potential to be a polynomial time reduction, though several things must be considered. The factorization of polynomials over finite fields runs (at best) in probabilistic polynomial time (see [7] for an overview of algorithms for factoring polynomials over finite fields) rather than in true polynomial time. The remaining steps of the algorithm do run in time that is polynomial in d, k, and $\log_2(p)$ (since most of these are simple linear algebra algorithms), and hence in all, when there is a unique family of solutions to $JX = XM$ of the form $kC_*^{-1}P_*^{-1}$ (where C_* is an arbitrary invertible matrix with zeros on all off-diagonals and P_* is an arbitrary permutation matrix), the reduction occurs in probabilistic polynomial time, whereby solving the STRKAP can be reduced to solving the DLP over the field $\mathbb{F}_{p^{kd}}$.

8 Conclusions and Future Research

It appears that the traditional approaches to attacking protocols over matrix groups are ineffective in the STRKAP due to the conjugating matrices A and B. Attempting to avoid these matrices by solving the DLP over the eigenvalues can, in some cases, lead to erroneous results, as the example in the appendix shows. Using eigenvectors to deduce information about A and B also seems to be a difficult approach because of the potential stretching and permuting induced by A and B.

Still, there are more avenues to try and more techniques to employ. For example, the action of $< Q >$ on $(\mathbb{F}_{p^k})^d$ by left multiplication induces a partition, while the action of $< AQ^jA^{-1} >$ on the same space yields a different partition, and it is possible that by studying these partitions that one can conclude enough information about A to reduce the problem to the DLP over the base field. Alternatively, there are other linear algebraic techniques that may be employed, such as using the characteristic polynomials of Q and AQ^jA^{-1} to deduce information about j.

Mathematical cryptology continues to be dynamic and challenging field. New key agreement protocols are being published frequently, as are new algorithms for decoding current protocols. These algorithms often utilize deep results from algebra and number theory, and may also help generate new ideas in these fields. Perhaps one of the most significant threat to current cryptographic protocols is the development of quantum computing; with the ability to simultaneously compute every power of a group element, key agreement protocols based on the discrete logarithm problem become very insecure. No one knows when, or even if, such computers will become feasible on the scale needed to disrupt modern communication. Until such systems become available, protocols such as the STRKAP will continue to provide the safest method of communication.

Acknowledgements

Dr David Yetter, KSU

Madeline Prenner, Wellesley College

Dr Conrad Plaut, UTK

Appendix

We will attempt to use the algorithm given in [4] to reduce the matrix DLP to the finite field DLP, and by example will show that these algorithms do not work with the STRKAP. To simplify matters, we will drop the requirement that the representation be irreducible; this cleans up the example a great deal, and should not affect the validity of the counterexample. Rather than constructing the infinite abelian group Γ and a representation into $GL_d(\mathbb{F}_{p^k})$, we will simply take a matrix $Q \in GL_d(\mathbb{F}_{p^k})$. Let $p = 11$ and $k = 1$, and let us consider the matrix

$$Q = \begin{pmatrix} 7 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix}$$

in $GL_d(\mathbb{F}_{11})$. Let us take Alice's exponent to be $a = 9$:

$$Q^9 = \begin{pmatrix} 8 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 7 \end{pmatrix}$$

Further, take the matrix A to be

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

which is simply a permutation matrix. Then the conjugate matrix $H = AQ^9A^{-1}$ is given:

$$H = AQ^9A^{-1} = \begin{pmatrix} 6 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

This is the matrix that Alice publishes. We now apply the algorithm presented in [4] in an attempt to find the exponent.

Let I be the $d \times d$ identity matrix. We first choose one of the eigenvalues of Q and a corresponding eigenvector; so choose the eigenvalue 7, and its corresponding eigenvector $[1, 0, 0, 0]^\intercal$ and take the matrix having this vector as its first column to be the identity matrix. Then the $(1, 1)$ entry $q_{1,1}$ of $I\,Q\,I$ has as its a^{th} power the $(1, 1)$ entry $h_{1,1}$of $I\,H\,I$, i.e $h_{1,1} = q_{1,1}^a$. In our case, we find then that $6 = 7^a$, which gives that $a \cong 7\ (mod\ 10)$. Repeating this for all eigenvalues, we find that a must equal 7, but in fact it is 9! Thus, this algorithm will give the incorrect answer, and moreover it will not notice that its output is incorrect!

Of course, this and similar algorithms were never meant to work on the STRKAP sort of protocol, so it is no surprise that they fail to give the correct value. The problem arises

because the conjugation can easily permute the order of the eigenvalues of Q, and therefore it becomes difficult to 'undo' this permutation without knowing A, and when Q has a complete set of eigenvalues which are all generators, it is truly impossible to use this protocol, as the above example shows.

References

[1] John R. Britnell and Mark Wildon, *On types and classes of commuting matrices over finite fields*, arXiv (2010).

[2] Povilas Tvarijonas Eligijus Sakalauskas and Andrius Raulynaitis, *Key agreement protocol (kap) using conjugacy and discrete logarithm problems in group representation level*, Informatica **18** (2007), no. 1, 115–124.

[3] F.R.Gentmacher, *Theory of matrices, volume 1*, Chelsea Publishing Company, 1960.

[4] A. J. Menezes, *The discrete logarithm problem in GL(n,q)*, Ars Combinatoria **47** (1997), 23–32.

[5] Andrew M. Odlyzko, *Discrete logarithms: The past and future*, Designs, Codes and Cryptography (2000).

[6] Carlos Cid Simon R. Blackburn and Ciaran Mullan, *Group theory in cryptography*, arXiv (2009).

[7] Joachim von zur Gathen and Daniel Panario, *Factoring polynomials over finite fields: A survey*, Journal of Symbolic Computation **31** (2001), 3–17.